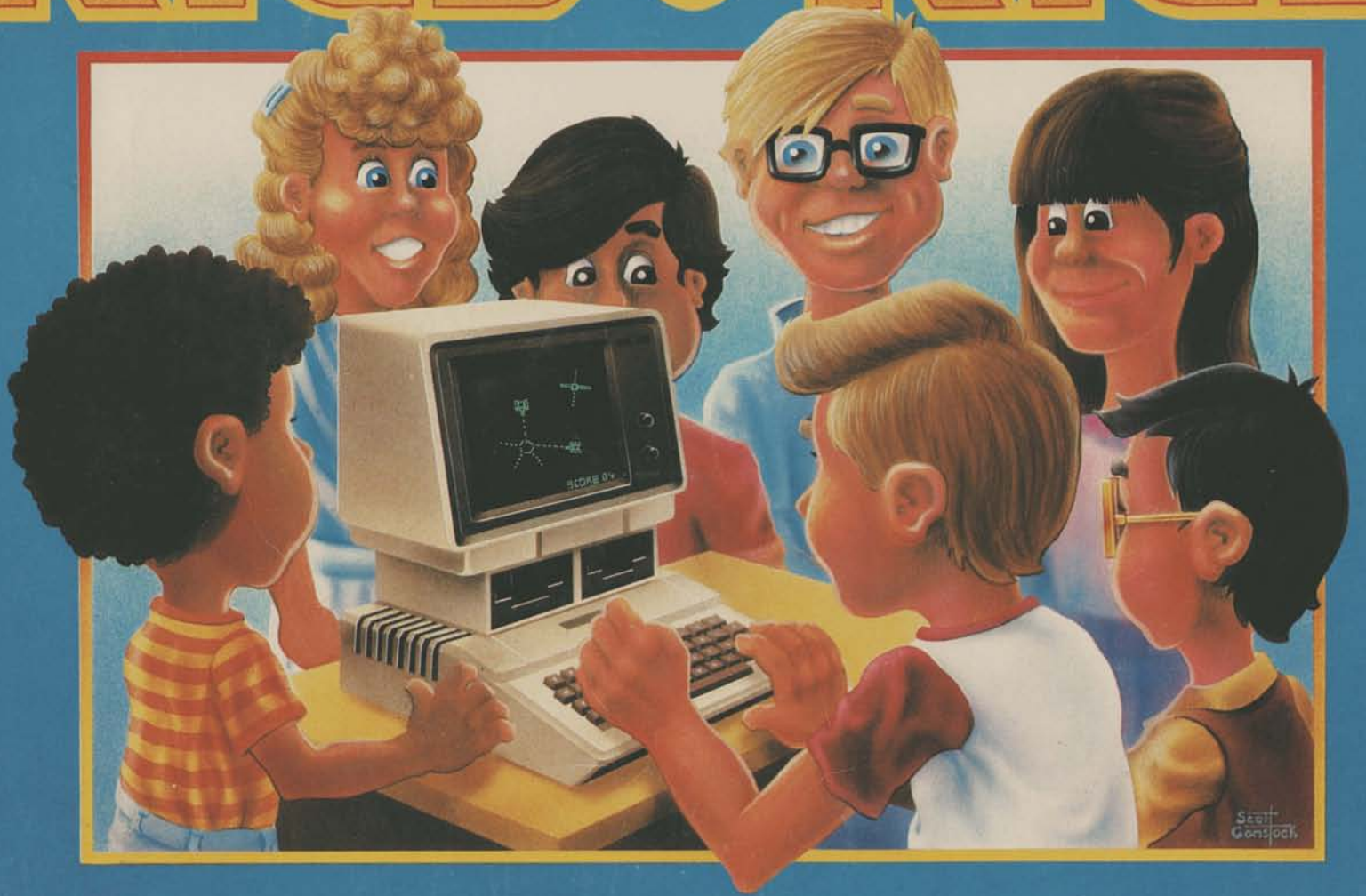
Kids to Kids

$9.95



ON THE APPLE COMPUTER

(for the Apple II, II+ & //e)

by
Billy Sanders
and
Sam Edge

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Kids to Kids

on the Apple Computer

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Kids to Kids

on the Apple Computer

(for the Apple II, II+ and //e)

by
Billy Sanders and Sam Edge

Illustrated by
Martin Cannon

Editor's Introduction
by
William B. Sanders, Ph.D.

DATAMOST™
8943 Fullbright Ave., Chatsworth, CA 91311-2750
(818) 709-1202

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

DATAMOST™

ISBN 0-88190-316-7

TABLE OF CONTENTS

EDITOR'S INTRODUCTION

When we brought our first computer home, our older son first learned how to run and program it. I explained the various parts, commands, and got him started programming with a promise of his own computer once he mastered the essential commands and statements. Like all nine year olds, his main interest was arcade games, and slowly he began wondering how he could make his own game. After some extensive work with the computer, he developed a rudimentary adventure game, being led by an interest in making the adventure rather than the pure joy I always felt in mastering the puzzle of programming.

Shortly after our older son became adept at working and programming his computer, his younger brother wanted to learn as well. I mumbled encouragement and went back to a matter engrossing my attention at the time and decided in the back of my mind to get around to helping him learn how to use the computer "soon." The next day I noticed that he was already using it without my "expertise." I asked him if his older brother knew about his using the computer, and he explained that it was his brother who had taught him how to work it. Later on, I found that the older brother could communicate with his younger brother in a way that was clear, concise and to the point.

Kids have always communicated better with one another than with adults, and this was simply another instance of that phenomenon. Also, kids seem to learn quickly from one another. Adults have ideas about what really interests kids, and sometimes they are right. However, kids just about always know about their mutual interests. And kids *never* want to be told by adults what is good for them. Since the passing on of valued knowledge from kids to kids has been going on since the first cave kid told another about the best tree to climb, why not apply

that process to learning about computers? After all, if one kid knows the "neatest" (if they still use that term) thing to do with a computer and how to program it, why not let him or her explain the whole process to another kid?

This book was born of the idea that kids can teach other kids better than anyone else. Adult intervention on my part was to give aid and assistance where required, and do all of the grimy work of formatting, correcting, and encouraging. To my surprise and delight, I was not overburdened with editorial tasks, as I had feared. The two young authors dove into their project with enthusiasm and ideas that seemed bottomless. Their energy level was always high, and discussions and phone calls concerning their opus occurred at all times of the day and night.

The level of the book is introductory, but the authors did include some more advanced topics near the end. This was done so that they could include some games and other programs they felt kids would enjoy. Otherwise, most of the material is designed to help kids get started using and programming their Apples.

The two authors, Billy Sanders and Sam Edge are 11 and 17 years old, respectively. They wanted me to include several people who helped them. First, Billy wants to thank Eric Goez for originally suggesting he create the book, his brother David, and his parents. Both Billy and Sam are grateful to Dave Gordon of DATAMOST for turning them loose in a room full of Apple games to play and lending them each an Apple computer to use for writing the book and programs. Sam would like to thank all his friends for their help, and he is also very grateful to his mother for all of her understanding support. Finally, as editor, I would like to thank Billy and Sam for doing such an excellent job.

William B. Sanders, Ph.D.
Series Editor

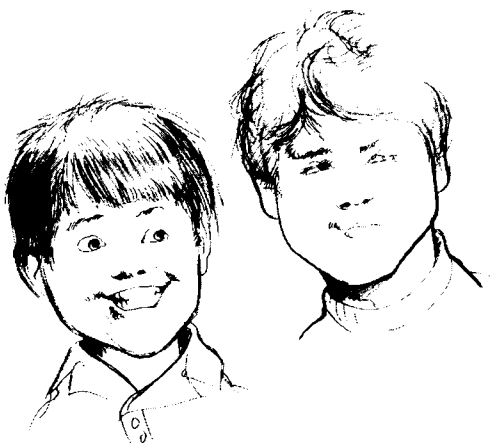
KIDS TO KIDS TO KIDS TO

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

1

GETTING SET

Hi, we're Billy Sanders and Sam Edge. We are kids like you (11 and 17 years old). That's why this book is called KIDS TO KIDS ON THE APPLE. It's a book for kids from two kids' points of view. We hope our perspective will help you understand how to use your Apple.



Let's get started by setting up your computer to your TV set. You will need an RF modulator, if you do not have one, you'll have to get one from your Apple dealer. With your RF modulator, there is a black switch box you connect to the back of your TV set. To do that, take the switch box wires and place them under the screws where it says VHF. Then just tighten the screws. Have you got it? Good. Second, you hook up the TV set to the computer. Your RF modulator goes inside the computer. Follow the directions that come with your RF modulator for the inside connection. (If you have a computer monitor, all you have to do is plug the video cable into the video port in the back of your computer and the other end into the back of the monitor. There is

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO

no need for an RF modulator for monitors.) Next, take out a long black cord which is in the box your computer came in. Hook one end of the cord into your RF modulator inside the computer, and the other end into the switch box. Now you have made the connection between your computer and your TV set. The instructions for your RF modulator will tell you which channel to use.

Now that you have hooked up the TV to the computer, plug the computer into the wall. You do that by taking the power cord and plugging it into the back of your computer on the left side, and plugging the other end into the nearest wall outlet.

Turn on your computer by flipping the little switch on the back left side of the computer. When you have done that, the following should appear on the top of the screen:

APPLE II

or

APPLE //e

This will be in the center of the screen. (The exact message will depend on when you got your computer and whether or not you have added any features to your computer.) If you

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

already have your disk drive attached, press the CTRL and RESET keys to make the disk drive stop spinning. When you are done with that, play with it for awhile. Press some different keys to see what happens.

Some of the keys you have are regular keys and some are irregular. You may be familiar with the keys from A-Z. Also, you know the number keys and their shift statements like !"$%&'()*:=−?/ >.<. But what about the up, down, left, and right arrow keys and the RETURN, CTRL, ESC, REPT, and RESET keys? Here is what they do. The up and down arrow keys move the cursor up and down the screen. (If you have an APPLE II or APPLE II+, there are no up and down arrows. You move the cursor up and down by first pressing the ESC key and then the I key for up and the M key for down. As soon as you press any other key, the I and M keys return to normal.) The same thing happens with the left and right arrow keys. What CTRL means is CONTROL. It can do certain things such as "break" a program. That will occur when you press CTRL C. Look below to see how this is done. Another one is REPT. Press REPT and any number or letter key, and it will make the key that you pushed repeat several times, depending on how long

you hold the REPT key down. (There is no REPT key on the Apple //e. By pressing and holding the key you want, the character will repeat.) The last key is the ESC key. That key is mainly used for editing, but there is another use for the key. When you press the ESC key you can move the cursor around the screen with the four letters IJKM. "I" moves the cursor up, "M" moves the cursor down, "J" moves it left, and "K" moves it right.

Pressing the RETURN key will end a program line and make the cursor jump a line. Press the CTRL-RESET key and that will interrupt the line that the program is on. For instance, if you are running a program and you press the CTRL and RESET keys, it will say BREAK IN 20 or some other number. Write this program to see how it works.

```
10 TEXT : HOME <RETURN>
20 PRINT "COMPUTER" <RETURN>
30 GOTO 20 <RETURN>
RUN <RETURN>
<PRESS CONTROL-C> or
<PRESS CONTROL-RESET>
```

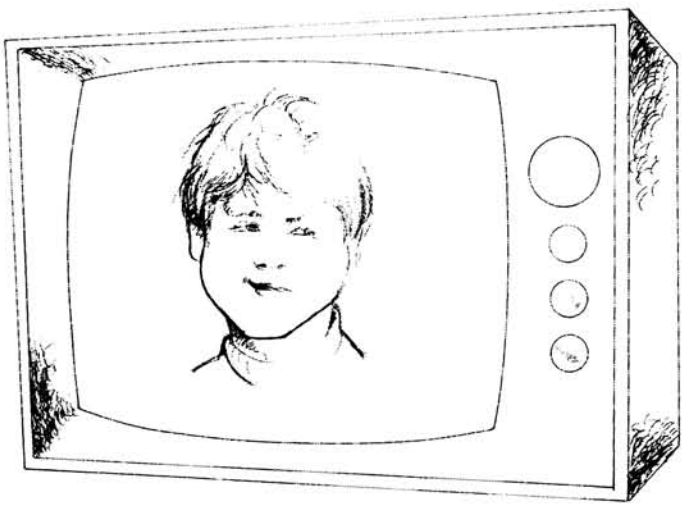
KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

By now you should have the computer hooked up to the TV and you should know what most of the keys mean. In the next chapter you will learn how to use the disk drive.

2

HOW TO USE THE DISK DRIVE

In this chapter you will be learning how to hook up and use the disk drive. If you want to use your disk drive, you have to hook it up first. Turn off your computer and TV. You will need a disk drive, a control card for the drive, and a cable to connect the drive to the control card. You SHOULD have bought this card with your disk drive. Attach this card to your disk drive cable. The first disk drive attaches to the top connector on the disk controller card. If you have two disk drives, attach the second cable to the bottom set of pin connectors on the same card.



The card that you attached to your disk drive has a flat portion which looks like "fingers" on the bottom. The "fingers" go into a slot in the inside of your Apple. Take the top off the computer and place the disk controller card in SLOT NUMBER 6. On the Apple //e, the leftmost slot is slot #1, but on the older Apple II and Apple II+s, it is slot 0. You have to count from 0

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Drives for your Apple!

instead of 1 to get the correct slot number. That's O 1 2 3 4 5 6 7 on the Apple II/II+. On the Apple //e, you count 1 2 3 4 5 6 7.

Now that you have connected your disk drive/s, let's make sure you can identify them. Your Apple dealer has stickers for your drives; one that says DRIVE 1, and another that says DRIVE 2. The sticker for DRIVE 1 goes on the drive connected to the TOP connector of your control card, and DRIVE 2 goes on the other drive.

Now that you have made the connection between your disk drive/s and your computer, let's INITIALIZE your diskette. What initializing means is to clear your diskette to get it prepared for saving and running programs. In computer talk, it "formats" the diskette. Here's how to initialize a diskette: (WARNING: Make sure you use a blank diskette for this exercise. If you use a diskette that has some files on it it will wipe out the entire diskette and the files will be erased.)

1. Boot your system with System Master (a diskette you should have gotten with your disk drive) and put in your blank disk.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

2. Type in the word NEW (clears everything out).

3. Write the following program exactly as it is!

```
10 TEST : HOME <RETURN>
20 PRINT"<YOUR NAME>'S DISK #1"
<RETURN>
30 END <RETURN>
INIT HELLO <RETURN>
```

Your disk drive will make some noises and after a while the noises will stop. Now the diskette is initialized and ready for use.

We are going to show you some useful commands which enable your disk drive and computer to talk with one another. First, we will learn how to see what's on a diskette. (Before you run a program you would want to see what is on the diskette, right? Right.) Here is how you do it: Type in the word CATALOG and press the RETURN key. Once you've done that, something like the following should appear on your screen:

```
*I HELLO
 A BUG ZAP
*B ALIEN
*A COLOR BOOK
```

Using the list above, here is how to run a program. Let's say the program that you want to run has an A or an I next to it. The A stands for APPLESOFT and the I stands for INTEGER. (If you boot your system with the System Master or an Apple //e or an Apple II/II+ with a language card installed in Slot 0, both I and A programs will work fine. This is because INTEGER BASIC is loaded on the language card and APPLESOFT BASIC is inside the machine in ROM (Read Only Memory). Otherwise, you may get a LANGUAGE NOT AVAILABLE message. If that happens, try to RUN only A files.) If it has an A or an I in front of it, all you have to do is type RUN, the name of the program, and press the RETURN key. If it has a B in front of it, it is a binary file. To run a binary file, all you have to do is type BRUN instead of RUN, and you will have your program/game.

Let's say you just wanted to look at the program and not use it. You do that by typing LOAD and the name of the program with A and I files. If you write in the word LIST and press the RETURN key, the program listing will appear on your screen.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Now that you know how to RUN programs, you should learn how they are SAVEd to diskette. This is how to SAVE a program. Once you have finished typing in a program just write SAVE, what you want to call your program, and press the RETURN key. That's a piece of cake. For example, try the following:

```
10 HOME <RETURN>
20 PRINT "APPLE COMPUTER"
<RETURN>
30 END <RETURN>
SAVE TEST <RETURN>
```

If you CATALOG your disk, you will see the file called TEST.

DELETEing a program is simple as well. All you have to do is type DELETE, the name of your program, and press the RETURN key. However, there may be a problem. Look back at the CATALOG and you will see that some of the programs have an * (asterisk) in front of them. The * means the program is LOCKed. You cannot DELETE a program that is LOCKed. To lock a program, just type LOCK, the name of your program, and press the RETURN key. If you want to delete a locked program you must

UNLOCK it. To unlock a program, all you have to do is type UNLOCK, the name of your program, and press RETURN.

One last word we would like to explain is RENAME. If you made a mistake while SAVEing a program, for instance, you SAVEd the program as GORK and you wanted PORK, all you have to do is type:

RENAME GORK,PORK <RETURN>

and that's it. The file named GORK will be replaced by PORK.

In this chapter you have learned how to do most of the basic things on the disk drive. In the next chapter you will be learning how to use PRINT and math statements.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

3

HOW TO USE PRINT AND MATH STATEMENTS

In this chapter you will be learning how to use PRINT statements, first with words and then with math statements. PRINT statements are very easy to use and they are the single most important programming statements. When you are using the PRINT statement you must always have quotes around the words you want printed to the screen unless you are using variables or numbers. (You will learn about variables in the next chapter.) Look at the sample below to get an idea about how a PRINT statement looks.

```
10 PRINT "HI MOM AND DAD."
20 PRINT "HOW IS EVERYTHING GOING?"
30 PRINT "EVERYTHING IS FINE OVER
   HERE, SON."
```

When you are finished typing the program, type RUN <RETURN>, and on your TV screen it will show,

```
HI MOM AND DAD.
HOW IS EVERYTHING GOING?
EVERYTHING IS FINE OVER HERE, SON.
```

The numbers 10, 20, and 30 are the line numbers. Every time you want to create a program, you must include line numbers. Line numbers can start at 0 and can be as high as 65000. Usually, you start a program with line number 10 and then add 10 to each line thereafter. The reason for leaving this gap between line numbers instead of increasing line numbers by one is because you may want to add program lines between existing lines. For example, if you answer the lines 1,2,3, and you want to add something between 1 and 2, you will have to start the program over again. If you number your program 10,20,30 as we did above, you could insert line 15 between lines 10 and 20. Maybe "insert" is not such a good word since the computer automatically places line 15 after line 10 and before line 20. Enough of line numbers, let's get back to PRINT statements.

A short way to tell the computer to PRINT is with the question mark. Type the program we just showed you, except instead of typing PRINT,

type just a question mark. Now when you type RUN (and <RETURN>), the program will run as though the question marks were PRINT statements.

Type in the next program to get used to using the PRINT statement. We will use the question mark this time. Also, we will introduce two new statements, TEXT and HOME. HOME will clear the screen of writing. TEXT will clear out any graphics and set the text window to full size. Lines 30 and 70 use PRINT statements without PRINTing anything. This will put a space (blank line) between the lines.

```
1Ø TEXT: HOME
2Ø ? "WHAT MONSTER WOULD YOU
   LIKE TO BE?"
3Ø PRINT
4Ø ? "1. DRAGON"
5Ø ? "2. OGRE"
6Ø ? "3. WEREWOLF"
7Ø PRINT
8Ø ? "YOU GUYS SCARE ME!"
```

When you RUN the program, it will look like this,


EVERYBODY CLEAR OUT!
I WANT THIS SCREEN EMPTY!
HOME

```
WHAT MONSTER WOULD YOU LIKE TO BE?

1. DRAGON
2. OGRE
3. WEREWOLF

YOU GUYS SCARE ME!
```

Before going on, we will learn another new statement, LIST. After RUNning your program, type in the word LIST. Your program will be LISTed on the screen as soon as you press RETURN. (Always press RETURN after an immediate command and after entering a program line. Words such as RUN and LIST are immediate commands.) Surprise! Instead of being question marks, the word PRINT has replaced them. The program will look like this on the screen.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 TEXT : HOME
20 PRINT "WHAT MONSTER WOULD
   YOU LIKE TO BE?"
30 PRINT
40 PRINT "1. DRAGON"
50 PRINT "2. OGRE"
60 PRINT "3. WEREWOLF"
70 PRINT
80 PRINT "YOU GUYS SCARE ME!"
```

For practice, try saving the program to disk. We will call the program MONSTER.

```
SAVE MONSTER <RETURN>
```

Now that you have learned how to PRINT letters, how about learning how to do arithmetic? Arithmetic is simple. All it is is adding, subtracting, multiplying, and dividing. (It is much more than that, but that's all we're going to do in this book.) We will begin with adding. Adding is the first thing a kid should know in arithmetic. Look at the example below to see how to do an addition statement.

```
PRINT 2+3 <RETURN>
```

After you press RETURN your screen will look like this:

```
PRINT 2+3
5
```

The PRINT statement tells the computer to PRINT the sum of 2+3 to the screen. If you put quotation marks around the 2+3, the computer would not PRINT the answer, instead it would look like this:

```
PRINT "2+3"
2+3
```

So remember, if you put quotation marks around something, it will PRINT whatever is inside the quotation marks; but if there are no quotation marks around numbers, it will PRINT the numbers or their arithmetic results.

Another thing a kid should know is how to subtract. Subtracting on your computer is just like adding. Instead of typing a plus sign, you would type a minus sign. For instance, look at the sample to see how a subtraction problem might look:

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
PRINT 23-18 <RETURN>
5
```

That was easy, but don't use your computer to get the answers to your homework! You will never learn how to do it in your head when you do not have your computer with you.

Multiplying works the same way as addition and subtraction. Try typing this on your screen:

```
PRINT 6 * 7 <RETURN>
42
```

When you are using your computer and you want to multiply, use the little star (called an asterisk) for the multiplication sign. You wouldn't use the X because to the computer that's all it is, the letter X. Division is closely related to multiplication. Division uses the slash mark that looks like this: /. You use the slash to figure a division problem:

```
PRINT 4/2 <RETURN>
2
```

It should say 2 right under the letter P. For some division problems where you get fractions, your computer will print up to eight digit decimals. When you get to junior high or high school, you will need all those decimal locations. The next program gives examples of all four math functions.

```
 10 TEXT:HOME
 20 PRINT "ADDITION PROBLEM: 28 + 49"
 30 PRINT "THE ANSWER IS "; 28+49
 40 PRINT
 50 PRINT "SUBTRACTION PROBLEM:
    83 - 47"
 60 PRINT "THE ANSWER IS "; 83-47
 70 PRINT
 80 PRINT "MULTIPLICATION PROBLEM:
    19 * 51"
 90 PRINT "THE ANSWER IS "; 19*51
100 PRINT
110 PRINT "DIVISION PROBLEM: 73 / 14"
120 PRINT "THE ANSWER IS "; 73/14
130 END
```

Notice that we were able to PRINT both the message THE ANSWER IS *and* the math problem on the same line. The single PRINT statement took care of printing both. We used the semi-colon (;) to put the two together on a single line.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

In this chapter you have learned how to use PRINT statements, addition, subtraction, multiplication, and division. You also learned how to clear the screen with HOME, LIST a program and use line numbers, and clear graphics out with TEXT. In the next chapter you will learn how to use variables.

4

HOW TO USE VARIABLES

NUMERIC VARIABLES

There are two kinds of variables we will show you in this chapter. The first one we'll see is called a NUMERIC variable. A numeric variable is like a slot where you keep a number. You name the slot with one or two letters and the name works just like a number. You can change the number in the slot and the letters that represent the slot will hold the new number. Look at the example to see how a numeric variable program might look:


```
10 TEXT: HOME
20 B=25
30 C=40
40 PRINT B + C
RUN <RETURN>
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

When you RUN the program, the screen will clear and 65 will appear. The reason it should say 65 is because 25 + 40 equals 65. The B in line 20 stands for 25. That's why it says B=25. On the next line it says C=40; so in the computer's memory that means the variable C stands for 40. The line below that, line 40, tells the computer to add B and C. It's the same as saying add 25 and 40 or PRINT 25 + 4Ø. Numeric variables are fun and easy to use. Following are two more programs you can use to see how NUMERIC variables work. Notice how we PRINT both messages and numbers (variables) in the first program. If we use the semicolon (;), whatever we PRINT will be next to the first item we PRINTed. Before you RUN the second program, see if you can guess what the computer will PRINT to the screen. These programs introduce a new statement, END. END tells the computer that it should stop. You do not need the END statement if it is at the very end of a program, but in some cases we will look at later, you do need it.

```
 10 TEXT: HOME
 20 A = 10
 30 A1 = A * A
 40 B = 20
 50 B1 = B * B
 60 PRINT "A = ";A
 70 PRINT "A1 = ";A1
 80 PRINT "B = ";B
 90 PRINT "B1 = ";B1
100 PRINT "A1 TIMES B1 =";A1 * B1
110 END
RUN <RETURN>
```

```
 10 TEXT : HOME
 20 X = 50
 30 Y = 60
 40 Z = 70
 50 A = X + Y
 60 B = X + Z
 70 C = Y + Z
 80 PRINT A
 90 PRINT B
100 PRINT C
110 END
RUN <RETURN>
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Write some other programs of your own and use numeric variables.

STRING VARIABLES

Another kind of variable we are going to present is called a STRING variable. A string variable is like a numeric variable except it has a dollar sign after it. A string variable looks like this: -> H$. String variables store "strings" in slots like numeric variables store numbers. A string is any message you put in quotation marks. For example, if you say A$ = "I'M A COMPUTER STAR", the message would be stored in the slot called A$. When you PRINT A$ your computer prints, I'M A COMPUTER STAR.

```
A$ = "I'M A COMPUTER STAR" <RETURN>
PRINT A$ <RETURN>
I'M A COMPUTER STAR
```

All string variables do is take something really big and change it into something small and easy to print. Look at the sample to see what a string variable might look like.

D$-DOG

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

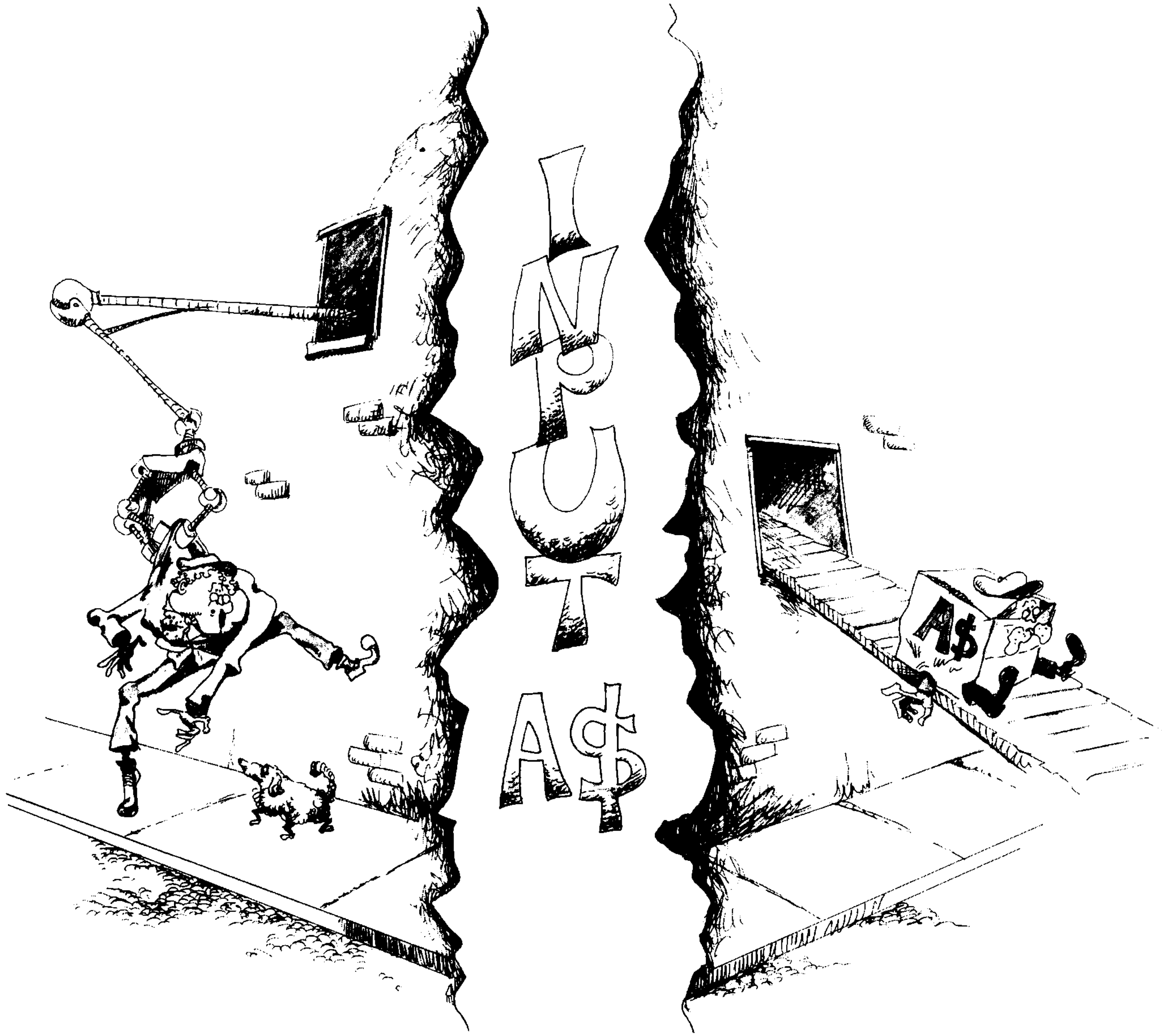
```
1Ø TEXT: HOME
2Ø A$ = "<YOUR NAME>"
3Ø PRINT "HI ";A$
4Ø END
RUN <RETURN>
```

When you store your name in the slot called A$ in line 20, it will PRINT your name when you PRINT A$ in line 30. We added HI and a semi-colon so that your computer would greet you. If you want, you can mix string and numeric variables in the same program. Look at the next program.

```
1Ø TEXT : HOME
2Ø AG = 11 : REM Input in your age
3Ø N$ = "<YOUR NAME>"
4Ø PRINT N$; " IS ";AG; "YEARS OLD"
5Ø END
RUN <RETURN>
```

INPUT STATEMENT

The last thing we are going to show you is how to INPUT variables. Again, this is very simple. Instead of defining a variable to equal something such as,


INPUT
A$
A$

```
A$ = "AIRPLANE"
```

we enter the name of the string or the value of the numeric variable after we RUN the program. For example, a program line asking for the variable content looks like the following:

```
10 INPUT A$
```

Whatever you enter will = A$. Now, let's look at a program using INPUT.

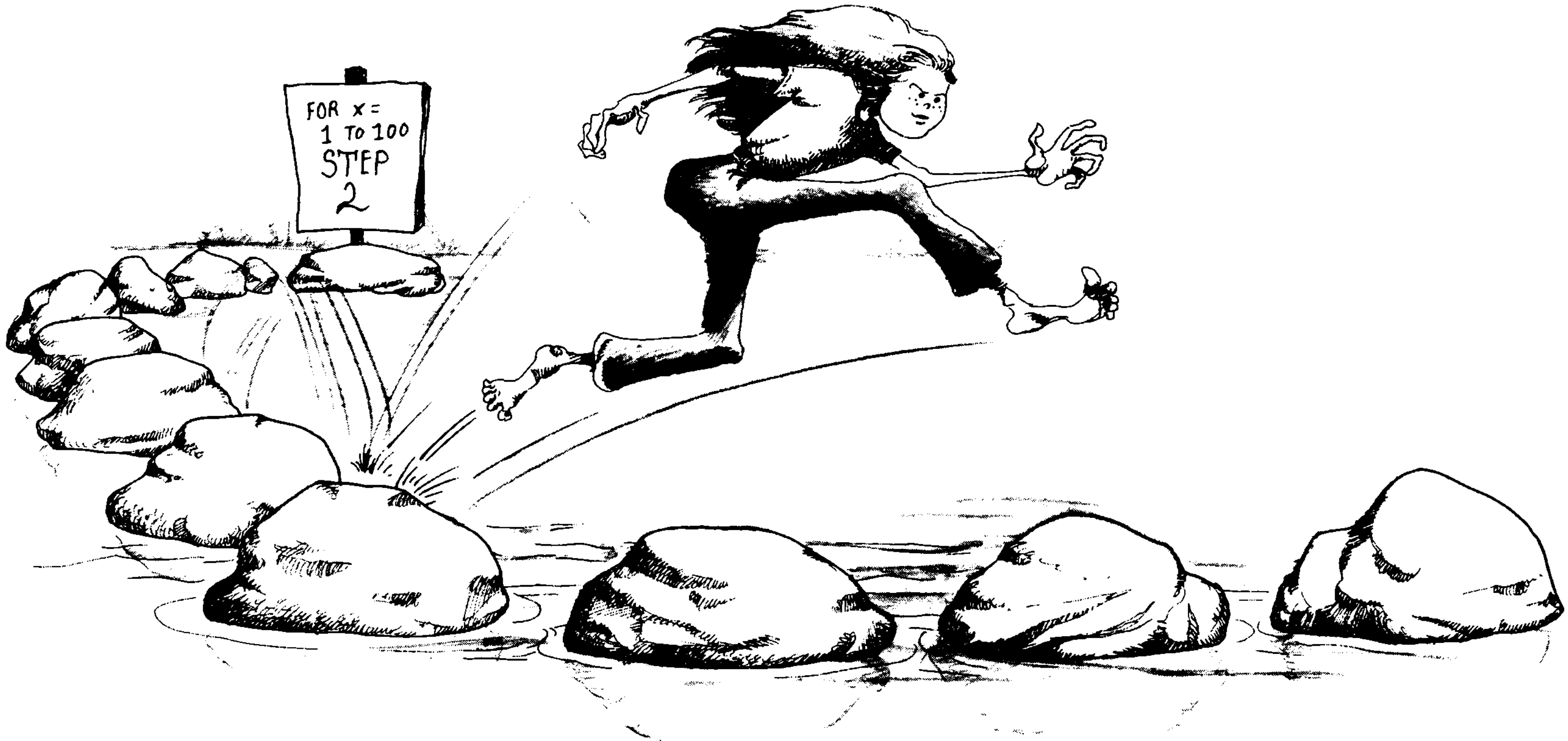
```
10 TEXT: HOME
20 PRINT "WHAT IS YOUR NAME?"
30 INPUT N$
40 PRINT "HOW OLD ARE YOU?"
50 INPUT AG
60 PRINT "HI ";N$
70 PRINT "YOU ARE " ;AG; " YEARS OLD"
80 END
```

You saw a similar program in this book, but by using the INPUT statement, we were able to enter any name and age we wanted. INPUT can change the value of the variables when we RUN the program. The next program shows how we INPUT and PRINTed two different strings for the same string variable.

```
10 TEXT : HOME
20 PRINT "ENTER WORD NUMBER ONE"
30 INPUT A$
40 PRINT A$
50 PRINT "ENTER WORD NUMBER TWO"
60 INPUT A$
70 PRINT A$
```

The first time we INPUT A$, it PRINTed the first word. The second time we INPUT A$ it printed the second word. That shows you that you can change the contents of a variable while a program is being RUN. Change the program so that you can add two more words. To do that, just add some more lines to the program beginning with line 80.

Now you should be able to see how important variables are. They are both useful and helpful. Well, guess what? This is the end of the chapter. You have learned how to use two kinds of variables: numeric and string variables. You have also learned how to INPUT variables. In the next chapter you will be learning how to use different kinds of loops such as the GOTO statement and the FOR/NEXT loop.


FOR x =
1 TO 100
STEP
2

5
USING LOOPS

There are two kinds of loops we will be discussing in this chapter. They are called the FOR/NEXT loop and the GOTO loop. First we will show you how to use a FOR/NEXT loop. Look at the example to see what a FOR/NEXT loop looks like. When you are finished looking at it, type it on your computer and RUN it to see what happens.


```
10 TEXT : HOME
20 FOR X=1 TO 10
30 PRINT X
40 NEXT X
50 END
```

In the second line, (line 20), you see that it says FOR X=1 TO 10. That means that X equals 1 to 10. The letter "X" is a type of variable. The value of X begins at 1 and goes up to 10. Each time the program hits the NEXT statement, it loops back to line 20 increasing the value of X by 1. This is why it is called a loop. The looping process continues until the value of X is 10; then it leaves the

loop and goes to the line after the statement NEXT. On line 30 it says to PRINT X. So the value of X will be PRINTed. The first time through the loop, the value is 1, then 2, then 3, and so on until it reaches 10. If the loop began with FOR X = 40 TO 50, it would PRINT 40, 41, 42, etc. until it reached 50. A good use for FOR/NEXT loops is when you have to do the same thing several times. If you have 10 names to enter, you could write a program that uses a loop from 1 to 10. Look at the next example.

```
10 TEXT : HOME
20 FOR X = 1 TO 10
30 PRINT "NAME: ";X
40 INPUT N$
50 NEXT X
60 END
```

If you did not have the FOR/NEXT loop, you would have had to do it the hard way. Look at the example of the hard way.

```
10 TEXT : HOME
20 REM THIS IS THE HARD WAY
30 PRINT "NAME 1 ";
40 INPUT N$
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
 5Ø PRINT "NAME 2 ";
 6Ø INPUT N$
 7Ø PRINT "NAME 3 ";
 8Ø INPUT N$
 9Ø PRINT "NAME 4 ";
1ØØ INPUT N$
11Ø PRINT "NAME 5 ";
12Ø INPUT N$
13Ø PRINT "NAME 6 ";
14Ø INPUT N$
15Ø PRINT "NAME 7 ";
16Ø INPUT N$
17Ø PRINT "NAME 8 ";
18Ø INPUT N$
19Ø PRINT "NAME 9 ";
2ØØ INPUT N$
21Ø PRINT "NAME 1Ø ";
22Ø INPUT N$
23Ø END
```

Using the FOR/NEXT loop it only took six lines to write the program. Using the "hard way" took 22 lines. (We didn't count the REM statement line. REM stands for REMark. All a REM does is to let you put comments in a program. It doesn't affect the program at all.)


OH, THAT'S JUST REM -- HE'S NOTHING BUT TALK!

You can also change the value in a FOR/NEXT loop with something other than one. Your computer can count by twos, threes or any other number you choose. To do this you have to use the STEP statement. It looks like this,

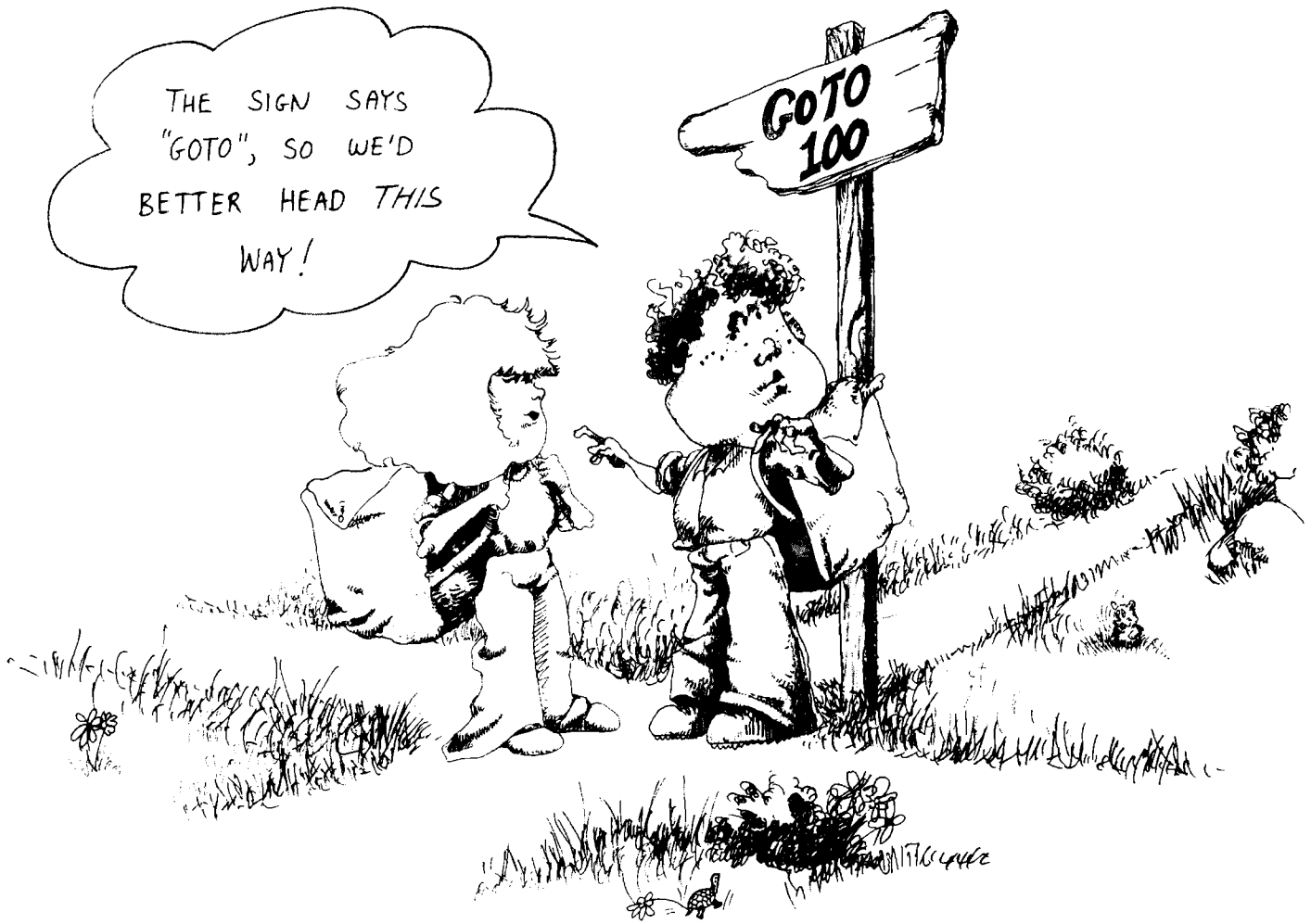
```
FOR X = 1 TO 100 STEP 2
```

Instead of counting from 1 to 100 by ones, it does it by twos. Enter the next program to see what happens when you use STEP.

```
10 TEXT : HOME
20 FOR X = 1 TO 100 STEP 2
30 PRINT X
40 NEXT X
50 END
```

Try changing the STEP value and see what happens. If you want to count backwards, use STEP and a minus sign (−). For instance, you could have,

```
FOR X = 100 TO 1 STEP -1
```



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Here's a program that will count from 100 to 1.

```
1Ø TEXT : HOME
2Ø FOR X = 1ØØ TO 1 STEP –1
3Ø PRINT X
4Ø NEXT X
5Ø END
```

Play with the statements to see what you will get. The comma (,) in line 30 will print the numbers in three columns. Try changing the comma to a semicolon (;) and a blank (PRINT X), to see the different results on your screen.

Now that you know how to use FOR/NEXT loops, let's see how well you can do with the GOTO statement. Look at the next program to see how GOTO might look in a program.

```
1Ø A$="FLOWER"
2Ø B$="BED"
3Ø PRINT A$;B$
4Ø GOTO 1Ø
```

On Lines 10,20, and 30 you see string variables which you learned about in the last chapter. Well, what it says is that A$ = FLOWER and

that B$, another string variable, = BED. And on line 30 it says to PRINT A$ and B$, so on your screen it would print FLOWERBED. On line 40 it says GOTO 10. What will happen is your program will go back to 10 and repeat everything over and over. Your screen will fill up with FLOWERBED. It will keep on doing that until you press RESET, CTRL C, turn it off, or until your computer breaks down. It is an endless loop. Endless loops are not used very much since you usually want your program to end at some point. Later on we will discuss a better use for the GOTO statement when we look at branching and decision making. But since we are talking about endless loops, the next program shows how to make a TV commercial with an endless loop.

```
 10 REM ****************************
 20 REM TELEVISION COMMERCIAL
 30 REM ****************************
 40 TEXT : HOME
 50 PRINT "SEE THE GREAT MOVIE"
 60 PRINT "THE CAT THAT ATE MIAMI"
 70 PRINT
 80 FOR W = 1 TO 1000
 90 NEXT W
100 GOTO 50
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

We used both a GOTO loop and a FOR/NEXT loop. The FOR/NEXT loop in lines 80 and 90 stopped the program for a couple of moments. The GOTO loop went back to line 50 and PRINTed the commercial message all over again. Try making your own commercial. For practice, try these next three programs.

```
1Ø FOR X=1 TO 1ØØØ
2Ø PRINT X
3Ø NEXT X
4Ø FOR V=1 TO 1ØØ
5Ø PRINT V,
6Ø NEXT V
7Ø PRINT: PRINT "IF YOU CAN SAY ALL
   THOSE NUMBERS IN ONE MINUTE
   YOU SHOULD BE ON THAT'S
   INCREDIBLE!"
```

```
1Ø PRINT "WHAT IS YOUR NAME?"
2Ø INPUT A$
3Ø PRINT A$
4Ø GOTO 3Ø
```

```
1Ø FOR I= 1 TO 5Ø
2Ø PRINT I
3Ø NEXT I
4Ø GOTO 1Ø
```

```
10 TEXT : HOME
20 FOR X = 1 TO 10
30 FOR Y = 10 TO 20
40 PRINT X,Y
50 NEXT Y
60 NEXT X
```

The last program contains a "nested loop." That's because the Y loop was "nested" inside the X loop.

In this chapter you have learned how to use two kinds of loops: FOR/NEXT and GOTO loops. In the next chapter you will learn how to use branching and subroutines. We will see more of the GOTO statements there.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS T

DS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

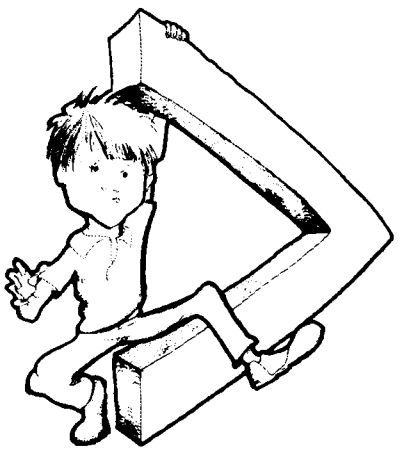
TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

6

DECISION MAKING

In this chapter we will explain and use the IF/THEN statement. It will enable you to do complex branching around your BASIC programs. There are two different types of branches or jumps; the GOTOs and the GOSUBs. These are direct branches.

We will start with the IF/THEN statement. This is called a "conditional branch" since it only will jump or branch under certain conditions. It is like saying, "IF you are going to walk in the rain THEN wear your raincoat." This means that you only have to wear your raincoat if you walk in the rain, but if you do not walk in the rain, you don't have to wear it. For example, in a program, the same statement would be as follows:



```
10 TEXT : HOME
20 PRINT "ENTER 1 FOR RAIN AND 2
   FOR NO RAIN"
30 INPUT R
40 IF R = 1 THEN GOTO 100
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


IF WE BELL THE CAT, THEN WE'LL HEAR HIM BEFORE HE ATTACKS!
-- AND IF YOU DON'T---
TRVTH

```
 50 IF R = 2 THEN GOTO 200
 60 END
100 REM **********
110 REM RAIN
120 REM **********
130 PRINT "WEAR YOUR RAINCOAT!"
140 END
200 REM *************
210 REM NO RAIN
220 REM *************
230 PRINT "YOU DON'T HAVE TO WEAR
    YOUR RAINCOAT."
240 END
```

Now, look at the next more general program and we will explain the branches with the IF/THEN statement.

```
 10 TEXT : HOME
 20 PRINT "ENTER 1 OR 2";
 30 INPUT N
 40 IF N = 1 THEN GOTO 100
 50 IF N = 2 THEN GOTO 200
 60 END
100 PRINT "YOU ENTERED ONE"
110 END
200 PRINT "YOU ENTERED TWO"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Lines 40 and 50 used IF/THEN statements. The statements checked if the value of N was equal to 1 or 2. IF the value of N was 1, THEN the program branched to line 100. IF the value of N was 2, THEN the program branched to line 200. Notice that we used two END statements. If you did not enter 1 or 2, then there was no branch, instead the program just ENDed at line 60. We also put an END statement at line 110, so that if you entered 1, the program would not continue to line 200 and PRINT the statement YOU ENTERED TWO.

The next program is more complex. It shows you the power you have using IF/THEN statements. It also introduces multiple statements on a single line. The RND function is explained below. To place multiple statements on a single line, we use a colon (:). It works like putting in another line number, but you can save memory and time by using it instead of a new line.

```
1Ø TEXT : HOME
2Ø PRINT "WOULD YOU LIKE ME TO
   TELL YOU A JOKE";
3Ø INPUT I$
4Ø IF I$ = "YES" THEN 1ØØ
5Ø IF I$ = "NO" THEN 7Ø
```

```
 60 GOTO 20 : REM *** GO ASK AGAIN ***
 70 PRINT : REM *** SKIP A LINE ***
 80 PRINT "SORRY TO HEAR IT, IT WAS
    A GOOD JOKE."
 90 END
100 PRINT "WHY DID THE FOOL DRIVER
    MAKE SEVEN PIT STOPS IN THE
    INDY 500?"
110 FOR A = 1 TO 1500 : NEXT A
    : REM *** WAIT A FEW SECONDS ***
120 PRINT : REM *** SKIP A LINE ***
130 PRINT "TWO FOR GAS AND FIVE
    FOR DIRECTIONS!!!"
140 END
```

In this program the computer asks you to INPUT whether or not you want to see a joke. You replied YES or NO. If your INPUT was YES, the computer branched to line 100 because of IF I$ = "YES" THEN GOTO 100. If you typed NO, line 40 would prove false and would therefore be skipped. Since line 50 asks the right question, the program branches to 70. Line 60 is just an error trap for type of wrong INPUT. Any INPUT other than YES or NO would cause the computer to jump back to line 20 and ask the question again. The next program is a simple game that will demonstrate the RND (RaNDom) function (which will be explained later in this chap-

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

ter), and also demonstrates some of the logical functions shown below (called relationals).

SYMBOL	MEANING
=	EQUAL TO
<	LESS THAN
< >	NOT EQUAL TO
>=	GREATER THAN OR EQUAL TO
<=	LESS THAN OR EQUAL TO

All these functions can be used with the IF/THEN statement to form complex decision making. Type in the following program and RUN it to see how they work.

```
10 REM ***
20 REM *** HIGH LOW GAME ***
30 REM ***
40 TEXT : HOME
50 PRINT "I AM THINKING OF A NUMBER
   BETWEEN Ø AND 1ØØ"
60 PRINT "YOU MUST TRY TO GUESS
   MY NUMBER BY TYPING IN
   DIFFERENT NUMBERS"
70 PRINT "AND I WILL TELL YOU IF
   YOU ARE TOO HIGH OR TOO LOW
   OR IF YOU HAVE IT"
80 RU = INT((RND)(1)*1ØØ+1)
90 PRINT : REM *** SKIP A LINE ***
```


RND

```
100 INPUT "ENTER GUESS"; G
110 IF G = RU THEN 180
    : REM *** CORRECT ***
120 IF G > RU THEN GOSUB 160
130 IF G < RU THEN GOSUB 170
140 GOTO 90
150 PRINT : REM *** SKIP A LINE ***
160 PRINT "IT WAS TOO HIGH" : RETURN
170 PRINT "IT WAS TOO LOW" : RETURN
180 FOR A = 1 TO 9 : PRINT CHR$(7)
    : NEXT A : REM *** SOUND LOOP ***
190 PRINT "*** YOU GOT IT ***"
200 PRINT : REM *** SKIP A LINE ***
210 PRINT "DO YOU WANT TO
    PLAY AGAIN"
220 INPUT I$
230 IF I$ = "YES" OR I$ = "Y" THEN 80
240 IF I$ = "NO" OR I$= "N" THEN 260
250 PRINT "WHAT?" : GOTO 210
260 PRINT
270 PRINT "OK, GOODBYE"
280 END
```

In this game you are asked to pick a number from 1 to 100. The computer will tell you if your guess is too high, too low, or "right on." In line 80, the variable RU is the random number between 1 and 100. It uses a function called RND.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

To get a range of random numbers, enter the highest random number which you want to be generated (in parentheses), after the multiplication sign. For example, if you wanted random numbers between 1 and 55, you would enter

```
INT(RND(1)*(55)+1)
```

The random numbers generated can be stored in variables, like we did with RU. Line 110 checks to see if the number you entered was equal to the computer's RANDOM number RU. If not, the program will continue to line 120 where, if the INPUT is greater than (>) the RANDOM number RU, the program will do a different branch called a GOSUB. When the computer is at line 160 it will do the following, 1) it will print out "IT WAS TOO HIGH", 2) it will see that there is a RETURN and the program will go back to where the GOSUB left off, at line 130. (Now wasn't that SIMPLE!) Even if we found that our number was greater than the RANDOM number, we still have to check to see if it is less than RU because the RETURN branched back to line 130, right after the last GOSUB. You might be able to figure out what's happening in 130. It's almost the same thing as in 120, except if your number was less than RU the GOSUB

would branch to 170. Again, after printing out the message IT WAS TOO LOW, the RETURN will go back to where to last GOSUB left off, line 140. (Line 180 is a sound loop which serves to congratulate you if your guess is correct.) Line 180 is where the computer branches to if your guess is correct in line 110. In line 220 you're asked if you would like to play again, to which you either reply YES or NO. Did you notice that in 230 the program checks to see if you INPUT YES or Y. That's where the OR function comes in. This function will let you compare two or more acceptable answers you wish to have. It's the same in 240 except it checks for NO OR N. And if none of the answers are acceptable, line 250 tells you that your answer was bad, and branches back to 210. Of course, we always have to say GOODBYE when people don't want to play anymore (lines 260-280).

There still might be some questions about the GOSUB/RETURN statements. If so, type the following and RUN it.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

*** IMPORTANT ***

THE FOLLOWING PROGRAM MUST BE SPACED EVENLY. IN THE PRINT STATEMENTS THERE ARE PERIODS IN PLACE OF SPACES. THIS IS FOR THE CONVENIENCE OF NOT HAVING TO COUNT THE SPACES IN THE LINE. FOR EXAMPLE, LINE 30 HAS 3 PERIODS (. . .). WHEN TYPING IN THE LINE YOU SHOULD INSERT 3 SPACES AND THEN THE MESSAGE.

*** EXAMPLE ***

(THE BOOK)

```
30 PRINT "...THE APPLE II MICRO
   COMPUTER"
```

(YOU TYPE)

```
30 PRINT "   THE APPLE II MICRO
   COMPUTER"
```

```
10 TEXT : HOME
20 GOSUB 140:
   REM *** PRINT 40 STARS ***
30 PRINT "......THE APPLE II MICRO
   COMPUTER"
40 GOSUB 140
50 PRINT : PRINT
```

```
 60 GOSUB 140
 70 PRINT "..... THIS IS A DEMO
    OF THE"
 80 PRINT
 90 PRINT "..... GOSUB/RETURN"
100 GOSUB 140
110 VTAB (18)
120 PRINT "..... < > < > THE
    END < > < >"
130 END
140 PRINT "*****************************";
150 REM ***
160 REM *** BRANCH BACK ***
170 REM ***
180 RETURN
```

Now let's discuss the flow of the program. In line 20 we GOSUB to line 140. Line 140 PRINTs out 40 stars and then goes down to line 180, where it branches back to the appropriate GOSUB. The program branches back and forth to the subroutine until it gets to line 130 and ENDs. Did you notice that instead of typing several rows of stars, you had to do it just once by using the GOSUB/RETURN routine? The most important thing about this is that we saved a great deal of typing and a bunch of valuable memory. The VTAB command was used to move

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

the cursor down eight lines. This method is cleaner than using a bunch of PRINT statements.

We finally made it! We're at the end of the chapter. Here are a few important things to remember. With the IF/THEN statements, you can do complex testing of either numeric or string variables. GOSUB/RETURN are good to use whenever a certain routine will be used more than once or twice.

7

ARRAYS AND READ/DATA

In this chapter we will discuss DIMs, ARRAYs, INPUTting ARRAYs, and how to use READ/DATA statements. The best way to think about ARRAYs would be to think of then as a kind of variable. To understand the use of an ARRAY, type in the next program. (REMEMBER TO CLEAR YOUR MEMORY WITH THE NEW STATEMENT.)

```
NEW <RETURN>
 10 TEXT : HOME
 20 PRINT "ENTER FIVE NAMES."
 30 INPUT A$
 40 INPUT B$
 50 INPUT C$
 60 INPUT D$
 70 INPUT E$
 80 TEXT : HOME
 90 REM ***
100 REM *** PRINT OUT THE NAMES ***
110 REM ***
120 PRINT "HIT RETURN TO SEE LIST
    OF NAMES."
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

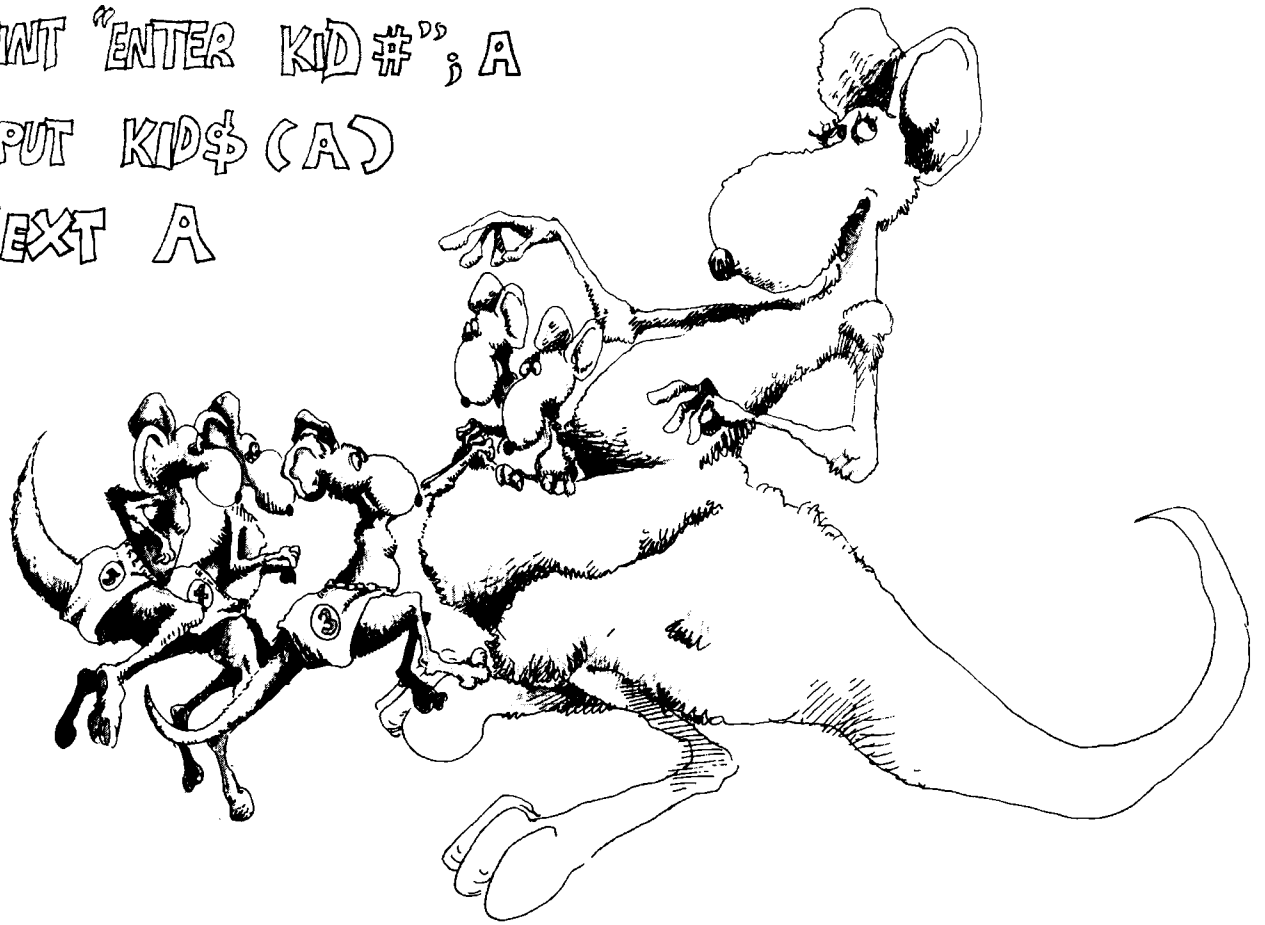
```
130 INPUT AA$
140 PRINT A$
150 PRINT B$
160 PRINT C$
170 PRINT D$
180 PRINT E$
190 END
```

The program should be easy to understand, but it's really not too good to use. If we have no more than one or two names or other types of INPUT data, this method would be fine. How about trying to INPUT 10 or 15 names? That would be a lot of INPUT statements! This is where arrays can help us. Try the following program:

```
NEW <RETURN>
 10 TEXT : HOME
 20 PRINT "ENTER FIVE NAMES"
 30 PRINT : REM *** SKIP A LINE ***
 40 REM ***
 50 REM *** INPUT 5 NAMES ***
 60 REM ***
 70 FOR A = 1 TO 5 : REM *** SET LOOP ***
 80 PRINT "ENTER NAME #" ; A;
 90 INPUT NAME$(A)
100 NEXT A
110 PRINT "HIT <RETURN> TO
    SEE NAMES"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 FOR A=1 TO 5
20 PRINT "ENTER KID#"; A
30 INPUT KID$(A)
40 NEXT A
```



```
12Ø INPUT ET$
13Ø PRINT : REM *** SKIP A LINE ***
14Ø REM *** PRINT OUT NAMES ***
15Ø FOR A = 1 TO 5
16Ø PRINT "NAME #" ; A ; "=" ; NAME$(A)
17Ø NEXT A
18Ø END
```

OK, now you might see the purpose of an array. If not, read carefully. In line 10 we set up the loop from 1 to 5. In 40, we ask for the string variable NAME$(A). At this point in the program, A=1; therefore, you're INPUTting NAME$(1). We will continue doing so until the loop is done (A=5). In line 90 we'll set up the loop again. Line 100 will PRINT the number we're on and then PRINT out NAME$(1), and continue until A=5 (or until we INPUT NAME$(5)).

Let's say that you want to add more than five names. You would adjust the loops to a higher number. For example, change FOR A = 1 TO 5 to FOR A = 1 TO 2Ø. Go ahead and change line 20 to 2Ø FOR A = 1 TO 15.

RUN IT !!!

You should have run into a problem (BAD SUBSCRIPT ERROR). When using strings or variables greater than 10, you must DIM

(DIMension) that particular string or variable. The usual way to DIM an ARRAY is to DIM it to the highest number of times we want to use the ARRAY. For example, to DIM the array NAME$ to 15 we would enter

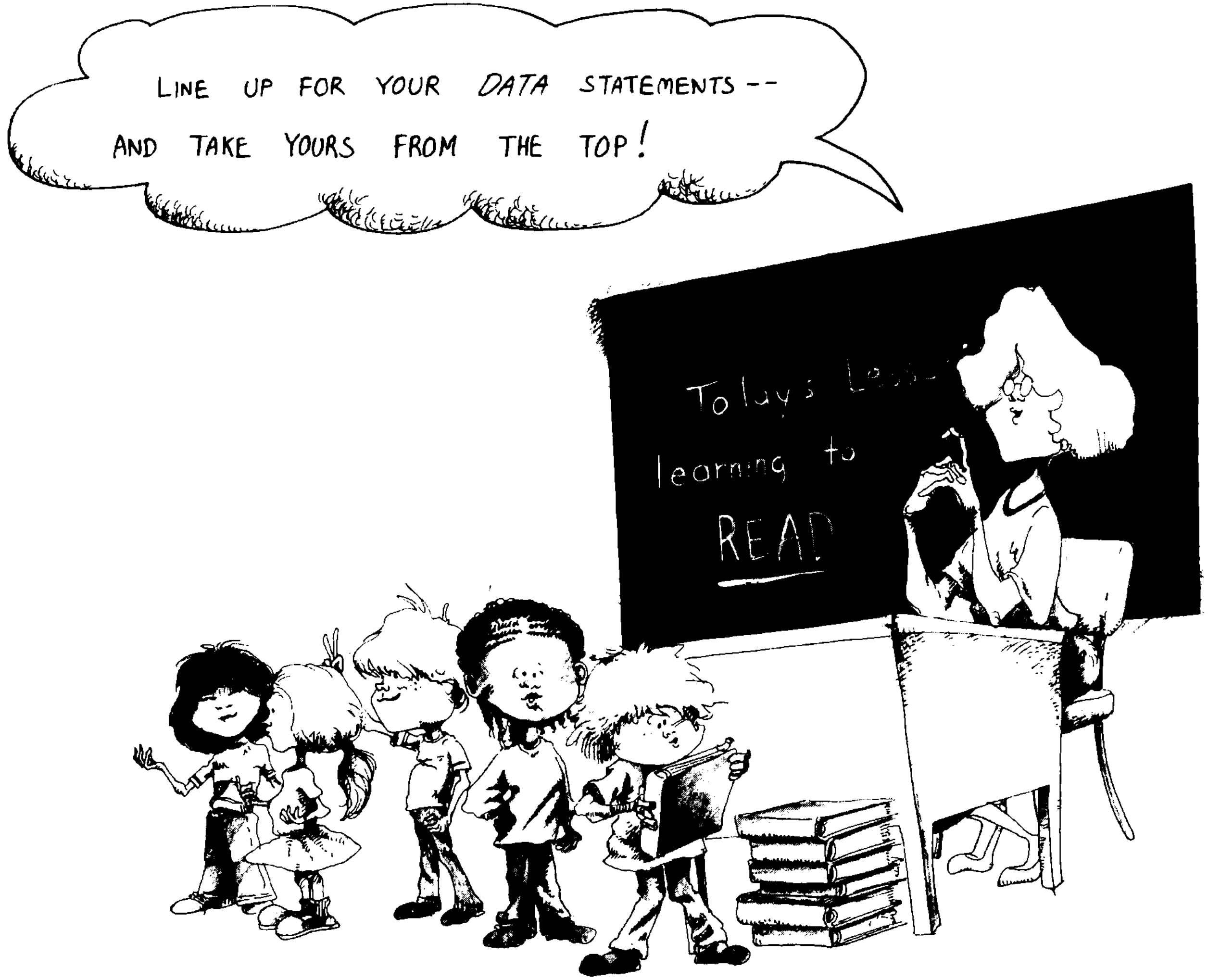
```
DIM NAME$(15)
```

This is because we will be using the string NAME$ 15 different times and the computer must make room for the ARRAY data. Remember that DIM reserves a certain amount of memory when you specify how much you need by DIMension. Let's fix the problem. Try this, insert the following line.

```
12 DIM NAME$(15)
```

You should have no trouble getting to the 15th INPUT. What you have now is 15 variable names, NAME$(1) to NAME$(15). Look at the following list below comparing regular string variables to array variables. We only used an array of four in the example, but you can get the idea of how much easier it is to use arrays in certain applications instead of variables.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


LINE UP FOR YOUR DATA STATEMENTS --
AND TAKE YOURS FROM THE TOP!
learning to
READ

Regular	Array
A$	A$(1)
B$	A$(2)
C$	A$(3)
D$	A$(4)

With arrays we can generate the variables names using FOR/NEXT loops as we did in our sample program. It saves a lot of time keying in variable names and makes our programs more flexible.

It's time we learn two more different BASIC statements. These next statements are the READ/DATA statements. Type in the following program and RUN it.

```
10 TEXT : HOME
20 FOR A = 1 TO 5
30 READ D$ : REM *** GET DATA ***
40 PRINT D$
50 NEXT A
60 DATA MONSTERS, GOBLINS, WITCHES,
   GHOSTS, VAMPIRES
70 END
```

The program first sets the loop from 1 to 5 in line 20. Then in 30 it READs the DATA from line 60 and stores it in D$. You must always separate each piece of DATA by a comma (,). This tells the computer where a new piece of data starts and ends. The first time through the loop the vari-

able D$ reads MONSTERS, then GOBLINS, and finally VAMPIRES. After the program is RUN, the values of D$ are as follows:

```
D$ = MONSTERS <-A =1
D$ = GOBLIN <-A =2
D$ = WITCHES <-A =3
D$ = GHOSTS <-A =4
D$ = VAMPIRES <-A =5
```

In the next program we're going to tinker with transferring data from DATA statements into string ARRAYs. This type of program would be useful when making a telephone/address file.

```
 1Ø TEXT : HOME
 2Ø C=1 : REM *** SET ARRAY POINTER
    TO ONE ***
 3Ø READ B$ : REM *** GET DATA ***
 4Ø IF B$ = "END" THEN GOTO 8Ø
 5Ø D$(C) = B$
 6Ø C=C+1 : REM *** INCREASE
    COUNTER OF ARRAY ***
 7Ø GOTO 3Ø
 8Ø PRINT : PRINT "HIT <RETURN>
    TO SEE NAMES"
 9Ø INPUT ET$
1ØØ REM ***
11Ø REM *** PRINT OUT NAMES ***
12Ø REM ***
```

```
130 FOR A = 1 TO C-1
140 PRINT "DATA #" ; A ; "=" ; D$(A)
150 NEXT A
160 END
170 REM ***
180 REM *** PLACE YOUR INFORMATION
    BELOW ***
190 REM ***
200 DATA DOG, CAT, COW, HORSE,
    PIG, GOAT, SHEEP, END
```

Notice we used the variable A in our FOR/NEXT loop in line 130 instead of C. It doesn't matter what variable name(s) we used since all we want it to do is to generate the numbers 1 to 7. That's because our array variables are actually D$(1) to D$(7) and not D$(C) or D$(A). The C and the A just represent different numbers. Also, note how we used END as the last element in our DATA statement. When the computer READ "END", it stopped READing DATA into the array and jumped to the routine for PRINTing the array to the screen.

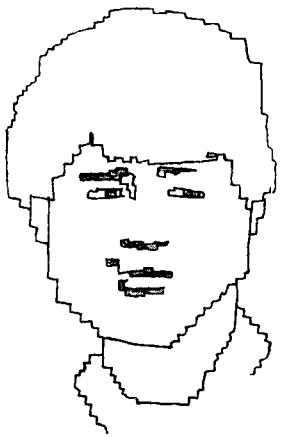
Since you've had so much experience with the DATA/READ statements, why not make that telephone/address file for yourself. (It's easy to do, but since your friends are probably some-

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

thing other than the barnyard characters we know, try putting names, addresses, and telephone numbers in your DATA statements. Use separate arrays to READ the different parts of a name/address/telephone number list.)

8

LOW RESOLUTION GRAPHICS



Hello, and welcome to the wonderful world of low resolution graphics. In this chapter you're going to learn how to use lo-res graphics on your Apple. Lo-res graphics are little blocks that can be made into bigger blocks which form an object or pattern. There are five new words you will be learning in this chapter. They are GR, COLOR, HLIN, VLIN, and PLOT. These are the five words you need to know in order to use lo-res graphics. Easy, isn't it? First, GR makes the computer go to lo-res GRaphics. Next, COLOR is used in programs for choosing the color you want your lines and plots to be. There are fifteen colors from which to choose (0-15) (gray is used twice).

0	BLACK
1	MAGENTA
2	DARK BLUE
3	PURPLE

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

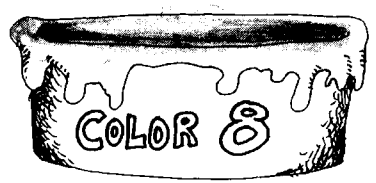
4	DARK GREEN
5	GRAY
6	BLUE
7	LIGHT BLUE
8	BROWN
9	ORANGE
10	GRAY
11	PINK
12	GREEN
13	YELLOW
14	AQUA
15	WHITE

NOTE: If you have a black and white screen, you may only want to use white. Experiment to see what the colors look like. All the other colors will be lighter than black and darker than white. If you have a color screen, you may want to use every color except black because when you write a program, the background is usually black and you cannot see black lines or plots. If you are using animation, black is good for erasing plots.

10 GR

20 COLOR=8

30 HLIN 5,35 AT 20



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

The next word you will be learning is HLIN. HLIN makes horizontal lines on your screen. Look at the next program to see how to use GR, COLOR, and HLIN.

```
10 TEXT:HOME
20 GR
30 FOR X = 1 TO 5
40 COLOR = X
50 HLIN 5,35 AT X+3
60 NEXT X
RUN <RETURN>
```

Let's start from line 10. TEXT : HOME sets your screen window and clears the screen. On line 20, GR sets your screen in the lo-res GRaphics mode. On line 30 you see a FOR/NEXT loop for X from 1 to 5. This will be used to get the colors from 1-5 on the screen. On line 40 you see the statement you learned earlier in this chapter, COLOR. It's color equals X, which is colors 1-5.

Next comes the tricky part. This is the big burrito; the guy who makes the lines go horizontally. Here's how HLIN works. After HLIN it says some numbers and two letters. Two of the numbers read as 5,35, that makes a line from 5 to 35

on your graphic screen at vertical position X+3. The first time, the X+3 equals 4, then 5, and on up to 8 at the end of the loop. We can better understand HLIN if we think of it as,

HLIN: begin line, end line, vertical position.

The following diagram shows how it works on the GRaphic screen.

LO-RES GRAPHIC SCREEN : HLIN

0 . . . . . 20 39

4 5----------35
5 5----------35
6 5----------35
7 5----------35
8 5----------35

.
.
.
.

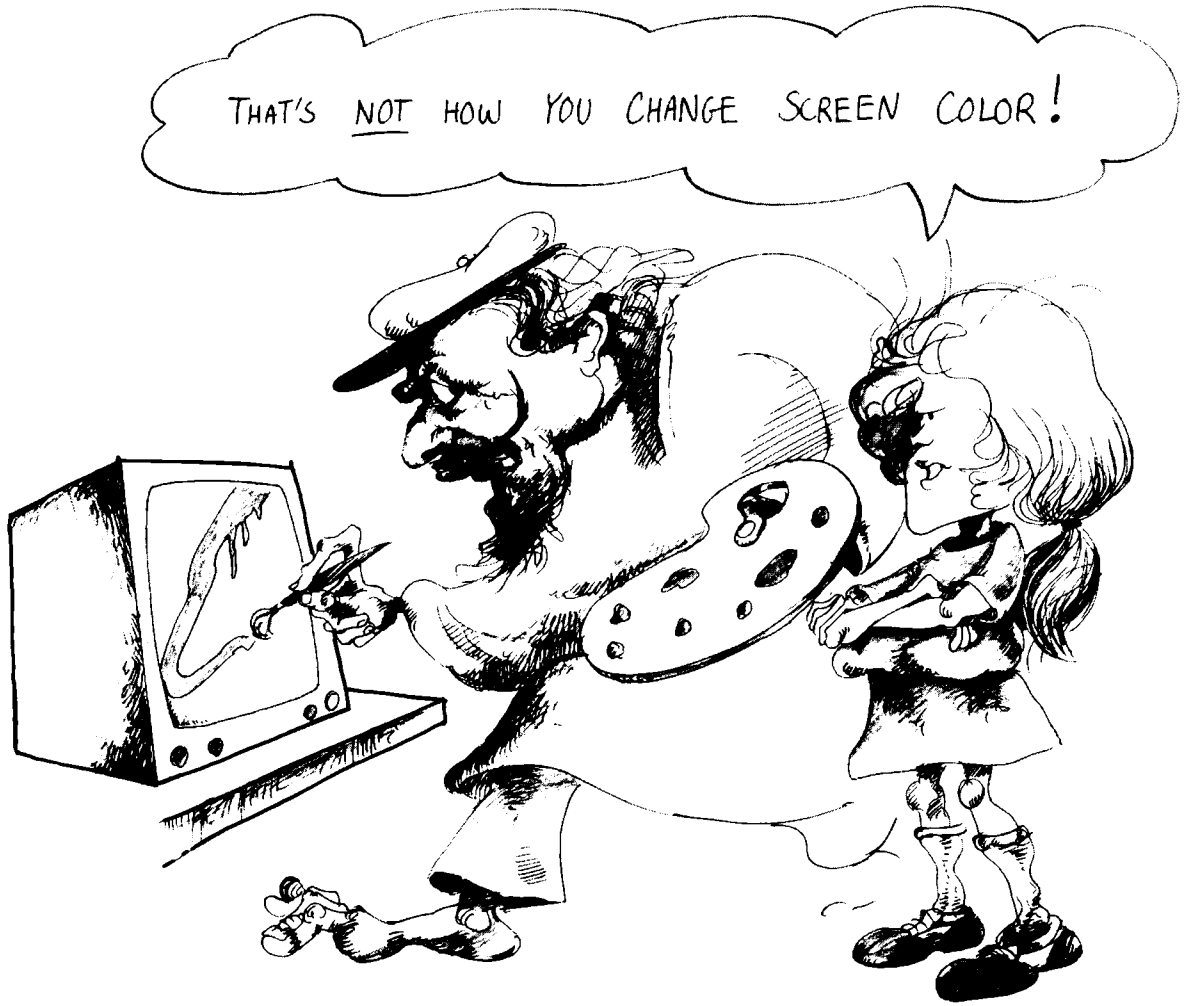
```
24
.
.
.
.
.
.
.
47 .....................
```

The loop generates five horizontal lines, each at a different vertical location since the vertical position is defined by the value of X in the FOR/NEXT loop.

Another word, VLIN, is similar to HLIN, but it makes vertical lines instead of horizontal lines. The format of VLIN is,

VLIN: begin line, end line, horizontal position

The only difference between VLIN and HLIN is that the last value in VLIN is for the horizontal position instead of the vertical position. Let's change our program so that it makes vertical lines instead of horizontal lines.



KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 TEXT:HOME
20 GR
30 FOR X = 1 TO 5
40 COLOR = X
50 VLIN 5,35 AT X+20
60 NEXT X
RUN <RETURN>
```

LO-RES GRAPHIC SCREEN : VLIN

```
0 . . . . . . . . . . 20 . . . . . . . . . . . . . . . . 39

.       55555
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       . . . . .
.       33333
35      55555

47 . . . . . . . . . . . . . . . . . . . . . . . . .
```

The last statement in this chapter is PLOT. You've learned how to make horizontal lines and vertical lines; now you are going to learn how to do single plots and diagonal lines. That's where PLOT comes in. PLOT doesn't make a complete line, all it does is make a little square dot; but with the dots in the right places, you've got a diagonal line. Look at the next programs to see how to use PLOT, first for individual blocks and then for diagonal lines.

```
10 TEXT : HOME
20 GR
30 COLOR = 15
40 PLOT 10,20
50 PLOT 20,10
60 PLOT 15,19
70 PLOT 1,1
80 PLOT 39,30
```

```
10 TEXT : HOME
30 GR
30 COLOR = 15
40 FOR K = 1 TO 10
50 INPUT "HORIZONTAL (0-39) ";X
60 INPUT "VERTICAL (0-30) ";Y
70 PLOT X,Y
80 NEXT K
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 TEXT : HOME
20 GR
25 COLOR=6
30 FOR X=1 TO 35
40 PLOT X,X
50 NEXT X
RUN <RETURN>
```

In the first two programs, individual plots are created using the horizontal and vertical positions. Using variables in the second program, you are able to PLOT all over the screen. In the third program, the FOR/NEXT loop generates a set of positions to line up the plots in a diagonal line. The first time through the loop, the horizontal and vertical values are 1 and 1, then 2 and 2, and so on until it reaches 35 and 35. See if you can make a diagonal line going the other way. (Hint: Use Step -1 in a loop.) Practice with low resolution graphics. Draw a face, your initials, a space fighter, a dragon, or your teacher!

DIAGONAL LINE

```
0. . . . . . . . . . . . . . 24 . . . . . . . . . . . . . . . . 47
 1,1
  2,2 <- Horizontal, vertical plot
   3,3
    4,4
     5,5
      6,6
       etc.
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

In the next chapter you will be learning how to do high resolution graphics. They are something like low resolution graphics except that the points are much finer and you can draw better pictures.

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

9

HIGH RESOLUTION GRAPHICS AND SOUND

In this chapter we will learn how to use the Apple's high resolution graphics and sound. One of the nicest features of the Apple is its fantastic graphics capability, which is one thing that has made the Apple so popular. Also, we'll see how to make sound and music using your Apple. Unlike the low and high resolution graphics where there are several words in BASIC to help, to make sound you have to rely on machine language routines. We will provide some routines that you can use in your own programs, and while you may not understand how they work, they will let you make more sound than you ever thought possible.

HIGH RESOLUTION GRAPHICS

To begin, there are basically two kinds of graphic

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

modes; (1) Low Resolution (explained in Chapter 8), and (2) High Resolution. To access the hi-res mode you use the HGR command. It's almost the same as setting the lo-res mode, but we place an H in front of GR to tell the computer we want the (H)igh resolution graphics mode. When in the HGR mode, the computer gives you these options: (1) The use of eight colors, (2) 280 by 192 matrix (this means there are 280 possible dots across, and 192 possible dots down), and (3) the use of the HPLOT command. Now, let's test our newly acquired knowledge. Type in the following:

```
1Ø TEXT : HOME
2Ø HGR
3Ø HCOLOR = 7
4Ø HPLOT Ø,Ø
5Ø END
```

You should see a single white dot in the upper left-hand corner of your screen. When first using HGR, we must specify which color we want by using the HCOLOR command.

NOTE: You only have to specify the HCOLOR *once, even after running another* HGR. *When*

using lo-res graphics, you must always re-specify the color. This is because the GR *command resets the color back to 0 (BLACK). The* HGR *does not reset the* HCOLOR, *but instead it uses the last color* HPLOT*ted for the new screen.*

The following list shows the high resolution colors and their related HCOLOR numbers:

(0) BLACK	(4) BLACK2
(1) GREEN	(5) GREEN2
(2) BLUE	(6) BLUE2
(3) WHITE1	(7) WHITE2

The colors such as GREEN2 (5) may vary. This is not the computer's fault; the problem lies within your color monitor or TV. You can adjust your set to acquire the desired colors. If you are using your home color TV, you may have to readjust the set for TV viewing.

Now that we've played with a single dot, let's generate some lines. Again we use the HPLOT command, but in a different form. Type in the following and run it. (Don't forget to clear memory with NEW before starting.)

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
10 TEXT : HOME
20 HGR
30 HCOLOR= 3
40 HPLOT 0,0 TO 279,191
50 END
```

There should be a white line from the top left-hand corner to the bottom right-hand corner of your screen. To help you understand this, here's what happened. The computer made the first plot at 0,0 (top left-hand corner) and then noticed that there was the TO command within the HPLOT line. This tells the computer to draw a line from the last HPLOTted point TO the next point. The next point to be HPLOTted is 279,191. Experiment a little by drawing lines from left to right, up and down, and anything else that comes to mind. When you're done, type in the following program and RUN it.

```
10 TEXT : HOME
20 HGR
30 HCOLOR = 5
40 HPLOT 0,0 TO 279,159 TO 0,159
   TO 279,0 TO 0,0
50 VTAB 22
60 HTAB 15
70 PRINT "HOUR GLASS"
80 END
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

In the last program we used multiple TO statements in a single HPLOT line. This can save a lot of memory and, most of all, it saves the hassle of typing in several other unnecessary lines.

Here are a couple of programs that deal with some high resolution graphics. The first is a program that projects multicolored spikes from the center to the outer edge of the screen. The program deals with some advanced math (you don't need to understand how it works at this point) and plots out a half circle with its other half to the side of it. This is called a sine wave. If you ever wanted to know how all these computer games get spiral and 3-D effects, this is how.

SPIKES

```
 10 TEXT : HOME
 20 HGR2
 30 REM *** X,Y IS MIDPOINT OF
    SCREEN ***
 40 X = 139
 50 Y = 96
 60 C = INT ( RND (1) * (7) + 1)
 70 X2 = INT ( RND (1) * (279) + 1)
 80 Y2 = INT ( RND (1) * (191) + 1)
 90 HCOLOR= C
100 HPLOT X,Y TO X2,Y2
110 GOTO 60
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

SINE WAVE

```
10 TEXT : HOME
20 HGR2
30 HCOLOR= 1
40 FOR Z = 0 TO 360
50 X = 0
60 X = Z * 278 / 360
70 Y = 96 - SIN (Z / 57.29578) * (191 / 2)
80 HPLOT X,Y
90 NEXT Z
```

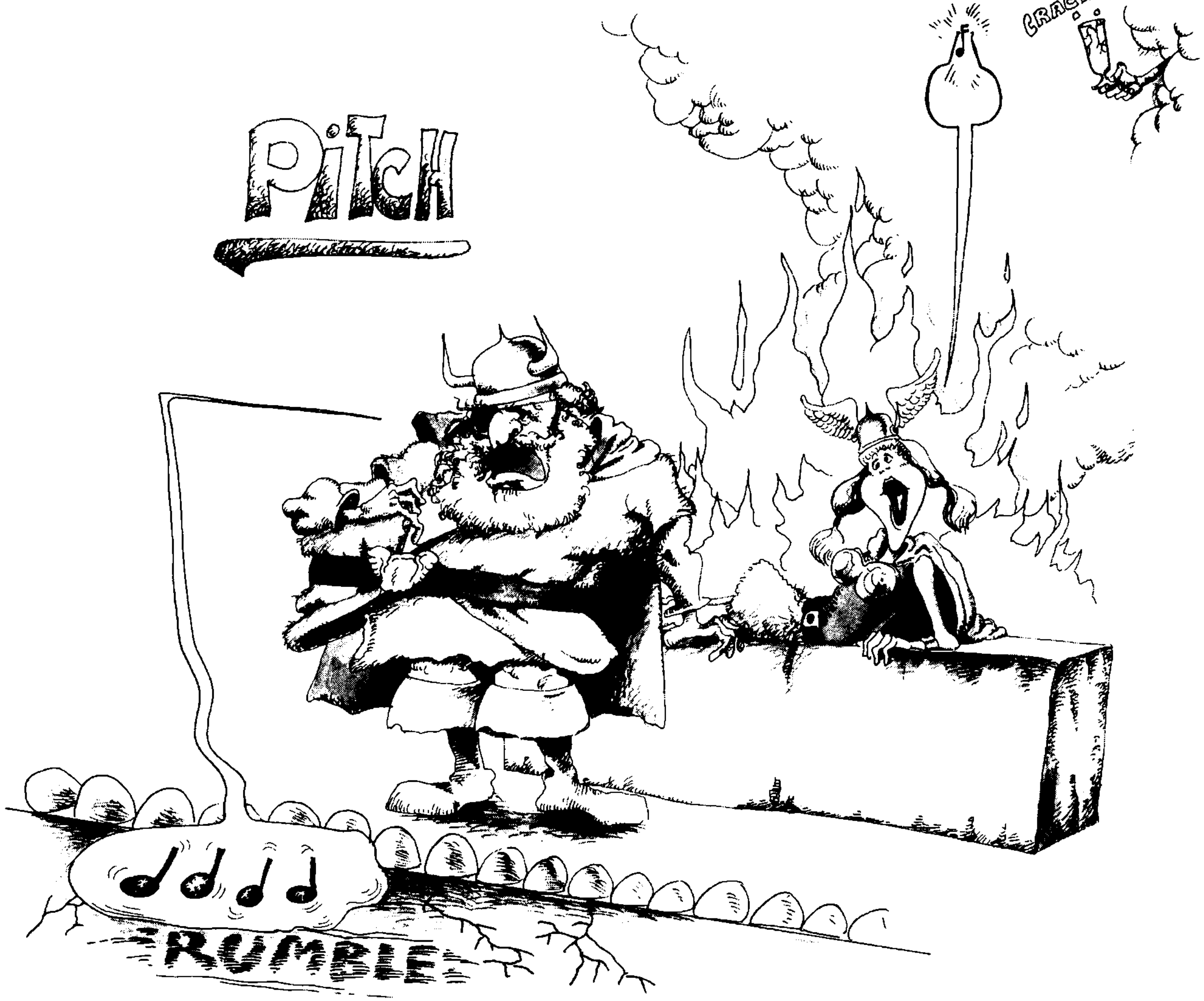
SOUND

The second half of the chapter is dedicated to the use of the Apple's speaker. Here we can produce computer sounds, even music! The actual production of sound is not explained because it is far beyond the limits of this book. We will only show you how to control the sound being produced, this is the most enjoyable part. Type in the following program and RUN it.

NOTE: It is very important that you correctly type in all the DATA *statements; otherwise, you can completely wipe out the program in memory. As in most cases, you should* ALWAYS *save a copy to disk or tape before* RUN*ning the program.*

```
100 TEXT : HOME
110 REM ***
120 REM *** GO AND PLACE SOUND
130 REM *** ROUTINE IN MEMORY
140 REM ***
150 GOSUB 5000
160 REM ***
170 REM *** T=TONE,D=DURATION
180 REM ***
190 D = 20
200 POKE 1,D
210 FOR T = 1 TO 255
220 REM ***
230 REM *** SET THE TONE
240 REM ***
250 POKE 0,T
260 VTAB (5)
270 HTAB (17)
280 PRINT "TONE=";T
290 REM ***
300 REM *** CALL MUSIC ROUTINE
310 REM ***
320 CALL 768
330 NEXT T
340 END
5000 REM *****************************
5010 REM *** MUSIC ***
5020 REM *** SUBROUTINE ***
5030 REM *****************************
```

PITCH



```
5040 REM
5050 REM
5060 REM
5070 FOR X = 768 TO 802
5080 READ D
5090 POKE X,D
5100 NEXT X
5110 RETURN
5120 DATA 165,0,201,255,240
5130 DATA 28,73,255,133,0,165
5140 DATA 1,72,173,48,192,136
5150 DATA 208,4,198,1,240,8
5160 DATA 202,208,246,166,0
5170 DATA 208,239,234,104,133
5180 DATA 1,96
```

If the program was typed in correctly, you should have heard a rising TONE. This is the entire scale that the Apple can generate (with this program). Using this program, the Apple can play from 1 to 255 different notes. When playing the notes, you can "hold" them for various lengths of time. This range is also from 1 to 255. In our last program, we played all of the notes from 1 to 255 with a length of 10 (10 is a very short duration). Look at the last program. In line 150 we sent the computer down to a subroutine which actually places this sound routine in your computer's memory.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Now we return from the subroutine and get to line 190, where we set the duration to 10. Line 200 places the value of the duration into memory. This is done by the function POKE. This function places a value in any available memory location. The memory location we want is 1. To put the value of D in 1 we would type POKE 1,D. We also do this for the value of T, but its memory location is 0. In line 210 we set up the TONE loop (1 to 255). Again, as before, we must POKE in the value of T. Now that we've specified everything necessary to produce a note, we must CALL the Sound Routine from memory. To accomplish this we use the CALL function in line 320. The CALL is used for RUNning or executing machine language routines that are in memory, like our sound routine. After playing the note, the computer will return to line 330 and continue looping until it reaches 255. There is one thing you should notice when using the different sound programs found in this chapter. You should notice that lines 5000-5180 are all exactly the same. To save yourself some typing, make a copy of lines 5000-5180 and SAVE them to tape or disk. Then, when you want just the sound routine, you can LOAD it up without any other program. Once the subroutine has been loaded into memory and you have RUN it,

you may type NEW to erase the unwanted BASIC program. Don't worry about losing the sound routine; it will be there until you turn off the computer. If you have done this, you may go ahead and develop your own music/sound programs. Remember the commands 1) POKE in the TONE at memory location 0 ; 2) POKE in the DURATION in memory location 1, and to CALL up the sound routine at 768; CALL 768. (If you don't remember what's happening, the routine will remember for you.)

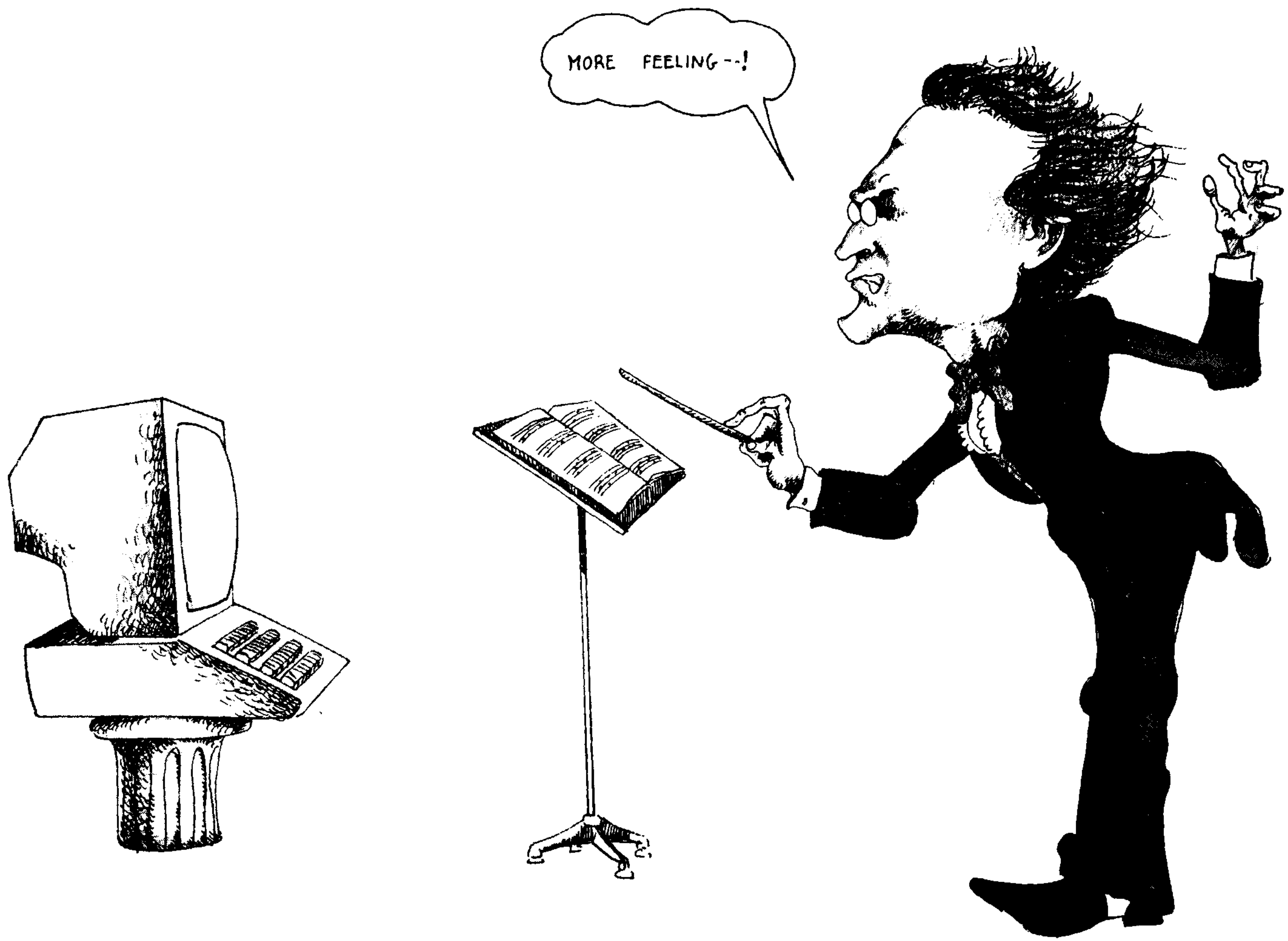
To put the sound routine in memory, RUN the following program before RUNning the others:

SOUND ROUTINE : SET-UP

```
5000 REM ***************************
5010 REM *** SOUND ***
5020 REM *** SUBROUTINE ***
5030 REM ***************************
5040 REM
5050 REM
5060 REM
5070 FOR X = 768 TO 802
5080 READ D
5090 POKE X,D
5100 NEXT X
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KID

)I SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK OT SDIK


MORE FEELING --!

```
5110 END
5120 DATA 165,0,201,255,240
5130 DATA 28,73,255,133,0,165
5140 DATA 1,72,173,48,192,136
5150 DATA 208,4,198,1,240,8
5160 DATA 202,208,246,166,0
5170 DATA 208,239,234,104,133
5180 DATA 1,96
5190 REM **********************
5200 REM RUN THIS BEFORE
5210 REM YOU RUN SOUND
5220 REM PROGRAMS
5230 REM **********************
```

CHARGE!

```
  10 REM ********************
  20 REM *** CHARGE ***
  30 REM ********************
  40 TEXT : HOME
  50 READ T,D
  60 IF T = 555 THEN FOR I = 1 TO D:
     NEXT I: GOTO 50
  70 IF T = 999 AND D = 999 THEN END
  80 POKE 0,T
  90 POKE 1,D
 100 CALL 768
```

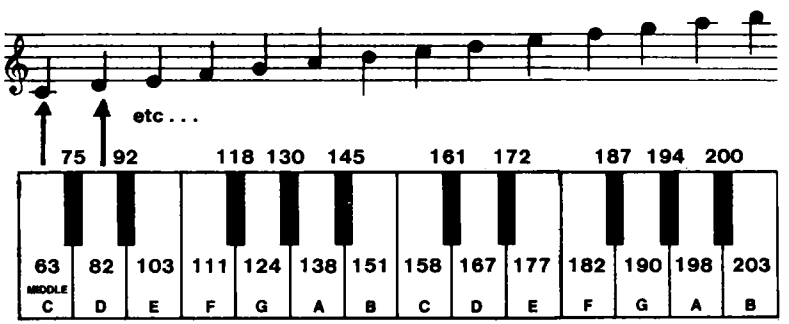
```
 110 GOTO 50
 120 DATA 63, 40, 111, 40, 141, 40, 159,
     50, 555, 100
 130 DATA 141, 60, 159, 255, 999, 999
5000 REM *** IF SOUND ROUTINE
5010 REM *** IS NOT IN MEMORY
5020 REM *** PLEASE DO SO.
```

If everything was typed in correctly, you should have heard the familiar tune CHARGE !!!. The program is very simple and straightforward to use. You can use this program or any other similar program that you develop to play such tunes. To understand this particular program, we'll have to do a little explaining about it first. In line 50, we get the first two pieces of data. The first piece of data is the TONE value, the second is the DURATION. Line 60 checks to see if T is equal to 555. If so, the computer will start a loop from 1 to D. This forms a timing loop for pausing in different parts of the song. Line 70 checks to see if T and D both equal 999, if so, the computer ends the program. If none of the above has happened, the computer POKEs in the TONE in memory location 0, and POKEs in the DURATION in memory location 1. Now we are all set to RUN the sound routine in memory by using CALL 768. Line 100 branches back to 50

and continues the same process until the computer ends at line 70. *Important: you must always place 999,999 at the end of your* DATA *statements so that the computer can end the program without an* OUT OF DATA ERROR.

Below is a note chart and the notes' values to be played on the Apple. Some of you more musically inclined individuals may feel that the note scale is off. The note chart was tuned-in by a Casio VL-Tone keyboard. You may want to upgrade the chart with a piano or some other finely tuned instrument.

NOTE CHART



To help make tuning your Apple easy, we've developed a utility to do this. Here's a description of the program. The program first asks for the TONE, you reply by giving it a note number that you want checked. You should hear the desired note. To tune in the Apple, you may use the I key to increase the note by one or hit the D key to decrease the note. When the note matches your instrument, you should write down the note for future use. One other thing we should

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
 10 TEXT : HOME
 20 INPUT "ENTER START NOTE ";ST
 30 POKE 1,255
 40 P = 0: IF PEEK ( - 16384) > 127
    THEN GOSUB 110
 50 POKE 0,ST
 60 VTAB (5)
 70 HTAB (2)
 80 PRINT "TONE=" ST
 90 CALL 768
100 GOTO 40
110 P = PEEK ( - 16384) - 128
120 POKE - 16368,0
130 IF P = 73 THEN ST = ST + 1
140 IF P = 68 THEN ST = ST - 1
150 IF P = 32 THEN GET A$
160 RETURN
```

mention is that once you have POKEed in the duration value, you don't have to re-POKE with the same DURATION until you need to change it. That about wraps it up for this chapter. Have fun with the programs and make some of your own!!! (DON'T FORGET TO RUN THE SOUND ROUTINE FIRST.)

Now that you know both types of graphics and sound, you can mix them together. See if you

can make a program that uses low or high resolution graphics and sound together. For example, you can make a program that will GOSUB to a different note each time a low resolution block is placed on a diagonal line, or make different sounds for different figures in high resolution graphics. In the next chapter we will see how to make a game. The techniques you learned in this chapter are used to make games.

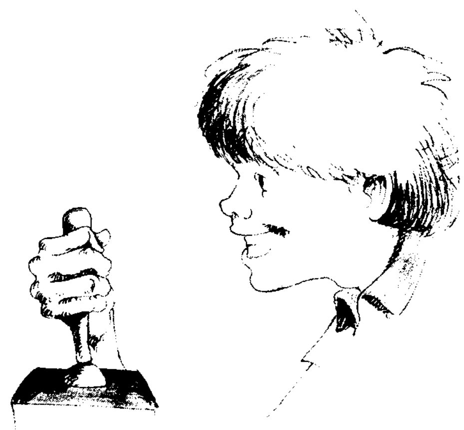
TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS T

10

HOW TO MAKE A GAME

In this chapter we'll cover how to put together a simple game. In developing a game, a major objective should be the purpose or goal of the game; such as catching bombs or shooting down the green invaders. It shouldn't be so hard that people won't play it because they always lose. Of course, it certainly should not be so easy that they can play for hours on end and always win. That's not very interesting either!

In the sample programs below, we will be using the PDL(x) statement. The way the PDL(x) statement works is as follows: The computer says that from the TOP to the BOTTOM there are 256 different positions, from 0 thru 63. This is also the same for the SIDE to SIDE position. Let's say that the joystick is in the middle, the computer will say the value is 128 . The way we get the value is by using the simple statement X = PDL(Ø). The variable X is now the variable of the SIDE to SIDE value of the PDL(x). Now we can get the value from the UP/DOWN value of



the PDL(x) by using Y = PDL(1). (In using game paddles, one paddle is PDL(0) and the other is PDL(1). To see which paddle is which, just define a variable as PDL(0). For example, if you put,

```
10 X = PDL(0)
20 HTAB 1 : VTAB 10 : PRINT X
30 GOTO 10
```

by turning the paddles, whichever one makes the values change is PDL(0). When you find which paddle is which, label them.) Look at the following example to learn how to use variables and your joystick.

```
10 TEXT :HOME : GR
20 COLOR = 9
30 X = PDL(0)/7 : REM *** GET SIDE TO
   SIDE ***
40 Y = PDL(1)/7 : REM *** GET
   UP/DOWN ***
70 PLOT X,Y
80 GOTO 30
```

Now, if you want to get an audio effect to demonstrate the PDL(x), and a clean screen when you press the fire button on the PDL(x), add the following lines:

```
 50 P = PEEK (-16287)
 60 IF P > 127 THEN 90
 80 GOTO 20
 90 PRINT CHR$ (7)
100 GR
110 GOTO 20
```

There is a special place for the fire button in the computer's memory. To see if the button is being pressed, we must look at that particular memory spot and see if the value is greater than (>) 127. If it is, the button has been pressed. The above explanation applies to lines 50 and 60. As you can see, if the button has been pressed, the computer will branch to line 90 where it rings the bell with the statement PRINT CHR$ (7), and clears the screen with GR. Notice that we branch all the way back to line 20. This is because after doing a GR the computer resets the color to zero (0, black).

Another neat addition to the program is as follows; change line 20 to read:


GOOD RESPONSE ···

```
20 COLOR = INT (RND(1) * (16) + 1)
```

Try experimenting to see what you can come up with. Come back here a little later and continue reading this chapter.

We're going to end this chapter with a simple game named LEMON DROP. The game is straightforward and easy to use. To control it you use the left/right arrow keys to move and the space bar to shoot. The controls are very tricky; to stop you must hit any key other than the three keys described above. You can vary how many lemons land before the game is over by changing line 670. At this point it is set to 9. Go to it and have a blast!!!

LEMON DROP

```
10 TEXT : HOME
20 XX = 15
30 HIT = 0
40 MISS = 0
50 GR
60 X = INT ( RND (1) * (35) + 1)
70 Y = 3
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
 80 GOSUB 400: REM *** PLOT
    THE BASE ***
 90 GOSUB 300: REM *** SCAN THE
    KEYBOARD ***
100 GOSUB 540: REM *** PLOT LEMON ***
110 IF Y > 37 THEN 650
120 IF FIRE < > 1 THEN 80
150 REM ****************************
160 REM *** FIRE ROUTINE ***
170 REM ****************************
180 FOR Z = 37 TO Y - 1 STEP - 1
190 COLOR= 15
200 PLOT XX,Z
210 IF SCRN( X,Y) = 15 THEN HIT =
    HIT + 1: GOTO 50
220 COLOR= 0
230 PLOT XX,Z
240 NEXT Z
250 FIRE = 0
260 GOTO 80
270 REM ****************************
280 REM *** SCAN KEYBOARD ***
290 REM ****************************
300 P = 0: IF PEEK ( - 16384) > 127
    THEN P = PEEK ( - 16384) - 128:
    POKE - 16368,0
310 K = PEEK ( - 16384)
320 IF K = 8 THEN XX = XX - 1
```

```
325 IF XX < 2 THEN XX=2
330 IF K = 21 THEN XX = XX + 1
335 IF XX > 37 THEN XX = 37
340 IF K = 32 THEN FIRE = 1
350 IF K = 81 THEN END
360 RETURN
370 REM ****************************
380 REM *** PLOT BASE ***
390 REM ****************************
400 COLOR= 0
410 PLOT XX - 1,38
420 PLOT XX + 1,38
430 PLOT XX - 2,39
440 PLOT XX = 2,39
450 COLOR= 6
460 PLOT XX,38
470 PLOT XX – 1,39
480 PLOT XX,39
490 PLOT XX + 1,39
500 RETURN

510 REM *****************************
520 REM *** PLOT THE LEMON ***
530 REM *****************************
540 COLOR= 0
550 Y = Y + INT ( RND (1) * (3) + 1)
560 PLOT X,Y – 1
570 PLOT X,Y – 2
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
580 PLOT X,Y - 3
590 COLOR= 13
600 PLOT X,Y
610 RETURN
620 REM ****************************
630 REM *** END THE PLAY ***
640 REM ****************************
650 PRINT CHR$ (7)
660 MISS = MISS + 1
670 IF MISS = 9 THEN GOTO 690
680 GOTO 60
690 TEXT : HOME
700 PRINT
710 PRINT "YOU SMACKED ";HIT;"
    LEMONS"
720 PRINT "AND MISSED ";MISS;" TOTAL"
730 VTAB (8)
740 INPUT "DO YOU WANT TO PLAY
    AGAIN (Y OR N)";A$
750 IF A$ = "YES" OR A$ = "Y" THEN
    GOTO 20
760 VTAB (15)
770 PRINT "OK, THANKS FOR PLAYING."
780 END
```

Well, we used a lot of different commands to create the game, and some are advanced. To make a good game, we sometimes have to get a little advanced. However, you have already learned most of the statements in the program, and if you study the program carefully, you should be able to get an idea how it works. With practice, you can make your own games.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO

11

HOW TO USE A PRINTER

Printers are like computer typewriters, but they can do a lot more than a typewriter. There is a special set of printer commands to learn, and that is what this whole chapter is about. We are going to show you how to do some things, like LISTing a program to your printer, and PRINTing a sentence or two (or three or four or more!).

Right now we are going to show you how to LIST a program to your printer. The command for this is LIST used in conjunction with PR#1. Look at the program below to see how LIST works with your printer. When you are finished looking at it, type it in.

```
10 PRINT "HI BILLY, HOW'S IT GOING?"
20 PRINT "FINE SAM, HOW'S IT GOING
   OVER THERE?"
30 PRINT "FINE, WELL I'LL SEE YOU
   TOMORROW"
40 PRINT "OK, BYE"
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


--LONG PROGRAM, FRED---

Now that you have finished the program, turn on your printer. Make sure it is ON LINE. If it is not ON LINE, flip the switch on your printer that will turn it ON LINE. (Some printers will have you choose SEL instead of ON LINE, but they are the same thing.) Now type:

```
PR#1
LIST
```

This will print out the entire program on the printer including the line numbers and the statements. It will print everything that is in the program. PR#1 turns on the printer, and LIST goes to the printer. Enter PR#Ø to get back to your screen. This may not seem too interesting, but when you are working on a long program trying to find bugs, it helps a lot to have a printed listing. It's called a "hard copy." Also, to send your friends a listing of your programs, the printed listing from your computer looks neat and you won't make mistakes copying it.

Now that you know how to print out a LISTing to your printer, we are going to show you how to print sentences, words, letters, or any other text you want to your printer. (We won't discuss

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

printing out graphics since that depends on the type of printer you have and is pretty advanced.) Look at the next program to see how to print text to your printer.

```
10 D$ = CHR$(4) : P$ = "I OWN A DOG.
   HIS NAME IS MACHO."
20 PRINT D$; "PR#1"
30 PRINT P$
40 PRINT D$; "PR#0"
```

If it didn't work, look at your printer and make sure it's turned on and ON LINE. On your printer it should say,

```
I OWN A DOG. HIS NAME IS MACHO.
```

The whole secret to PRINTing to your printer instead of to your screen is in line 20. The statement PRINT D$; "PR#1" sends what would normally go to your screen to your printer. The D$ is equal to Control-D, but instead of trying to print Control-D, which is invisible in a program listing, we used CHR$(4) which is the same thing. Whenever you issue a command to the printer or disk drive from a program, you have to use PRINT Control-D or, as we did, define D$ =

CHR$(4). In this case P$ in line 10 is I OWN A DOG. HIS NAME IS MACHO., and that is what is sent to the printer. Now that we have that out of the way, feel free to change P$ to whatever you want it to be. You can make a sentence or a paragraph or two.

Right now we will show you how to how to print lowercase letters on the Apple //e. Lowercase letters are simple; look at the example below to see how to create lower case letters on your printer.

```
10 J$ = "My best friend's name is Jonny"
20 PRINT D$; "PR#1"
30 PRINT J$
40 PRINT D$; "PR#0"
```

Make sure your printer is turned on and ON LINE. Now type RUN, and on your printer it should say,

My best friend's name is Jonny.

The only two letters that should be capitalized are the M in My and the J in Jonny. What you do to get lowercase on the Apple //e is to take it off

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

CAPS LOCK and press a key. Uppercase letters are made by pressing the SHIFT KEY and the letter you want. Each time you press the CAPS LOCK key, you toggle the upper and lowercase letters.

Our next program is a simple one which turns your printer into a typewriter. When you RUN the program it will clear the screen and put a question mark on the left side of the screen. Type in messages, and every time you hit RETURN, the text will be printed to your printer. Be sure not to put in too many letters — about 200 — before you press RETURN. You can use it to write letters to your friends. When you are finished, press # when the question mark appears and it will end.

KID PROCESSOR

```
10 TEXT : HOME : D$ = CHR$(4)
20 INPUT S$
30 IF S$ = "#" THEN END
40 GOSUB 100
50 PRINT
60 PRINT S$
70 PRINT D$; "PR¢0"
```

```
 80 GOTO 20
100 REM **************************
110 REM TURN ON THE PRINTER
100 REM **************************
130 PRINT D$;"PR#1"
140 RETURN
```

In this chapter you have learned how to make a printout of a program, print sentences out on the printer, and how to print out lower case letters on your printer. In the next chapter you will learn how to do your homework on your computer.

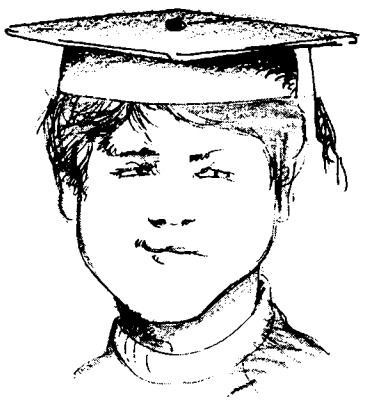
KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS

12

DOING HOMEWORK ON THE COMPUTER

In this chapter we'll show you how you can do your homework, so even your parents will want you to use your computer when doing homework. You say that's impossible? Well, type the following program and you'll see why it's not impossible. *(NOTE: Where you see periods in the program listing, put spaces.)*



```
10 HOME
20 PRINT " ..... APPLE PROBLEM
   CALCULATOR"
30 VTAB (8)
40 PRINT " ..... <1> .. ADDITION"
50 PRINT " ..... <2> .. SUBTRACTION"
60 PRINT " ..... <3> ..
   MULTIPLICATION"
70 PRINT " ..... <4> .. DIVISION"
80 PRINT " ..... <5> .. QUIT"
90 VTAB (20): HTAB (12)
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
100 INPUT "PICK A FUNCTION ";A
110 HOME
120 IF A = 1 THEN A$ = "ADDITION"
     : GOTO 210
130 IF A = 2 THEN A$ = "SUBTRACTION"
     : GOTO 210
140 IF A = 3 THEN A$ = "MULTIPLICATION"
     : GOTO 210
150 IF A = 4 THEN A$ = "DIVISION"
     : GOTO 210
160 IF A = 5 THEN 410
170 GOTO 10
180 REM *****************************
190 REM *** MAIN ROUTINE ***
200 REM *****************************
210 VTAB (3): HTAB (15): PRINT A$
220 VTAB (7)
230 INPUT "ENTER THE FIRST NUMBER ";B
240 VTAB (10)
250 INPUT "ENTER THE SECOND
     NUMBER ";C
260 VTAB (14)
270 REM *****************************
280 REM * CALCULATE ANSWER *
290 REM *****************************
300 IF A = 1 THEN D = B + C
310 IF A = 2 THEN D = B - C
320 IF A = 3 THEN D = B * C
```

```
330 IF A = 4 THEN D = B / C
340 INVERSE
350 INPUT "WHAT IS YOUR ANSWER ?";A
360 IF A < > D THEN 420
    : REM *** WRONG ANSWER ***
370 PRINT "YOU'RE RIGHT!"
380 VTAB (18): NORMAL
390 INPUT "DO YOU WANT TO TRY
    ANOTHER?";A$
400 IF A$ = "YES" OR A$ = "Y" THEN 10
410 PRINT : PRINT "OK, GOODBYE": END
420 PRINT "SORRY, THE ANSWER IS ";D
430 GOTO 380
```

You should notice that the program is very short. We thought there would be no need to write four almost identical programs. We could have made the program from three main GOSUB routines combined into one main program, but instead we did it a little differently.

Line 210 is the beginning of the main routine which asks for both numbers. You can also refer to the numbers as the NUMERATORS, which are the top or first numbers, and the DENOMINATORS, which are the bottom or the last numbers. This really isn't too important right here, but it is good to know. Your parents should

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

really appreciate a program like this (especially if they have trouble helping you with your math homework). If you want to be a real whiz kid, why don't you try to make a spelling bee program? It will guarantee a 100% on your next test, provided that you study with the program. Here's a little program that will help you get started, see if you can improve it. (Notice the new words FLASH and NORMAL. The word FLASH makes your characters flash. NORMAL puts everything back to the regular way of displaying characters. Also try using INVERSE instead of FLASH.):

```
 10 TEXT : HOME
 20 INPUT "HOW MANY SPELLING
    WORDS"; N
 30 HOME : DIM W$(N)
 40 FOR X = 1 TO N
 50 PRINT "WORD #";X;
 60 INPUT " "; W$(X)
 70 NEXT X
100 REM ********************
110 REM SPELLING TEST
120 REM ********************
130 FOR X = 1 TO N
140 HOME
150 FLASH : REM MAKES WORD FLASH
```

```
160 VTAB 10 : HTAB 10
170 PRINT W$(X)
180 FOR PAUSE = 1 TO 500 : NEXT PAUSE
190 NORMAL : HOME
200 VTAB 10
210 INPUT "SPELL THE WORD ";WD$
220 IF WD$ = W$(X) THEN PRINT: PRINT
    "YOU GOT IT RIGHT!" : GOTO 250
230 PRINT "THAT'S NOT QUITE IT.
    TRY AGAIN"
240 FOR PAUSE = 1 TO 1000
    : NEXT PAUSE : GOTO 200
250 FOR PAUSE = 1 TO 1000 : NEXT PAUSE
260 NEXT X
300 HOME
310 PRINT "THAT'S ALL. GOOD LUCK
    ON YOUR TEST"
320 END
```

WoRdPrOcEsSiNg

Have you ever wished you could be a fancy typist and have great looking school reports? Well wish no more. The power is right at your fingertips (with the help of your Apple computer). Most computers are capable of using a word processor. For the Apple computer there

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

are several different ones from which to choose. Here are just a few: Bank Street Writer (Broderbund Software) was designed with kids in mind. It is easy to use, it has just about everything you would need for school. Apple Writer II (Apple) comes with a lot of Apple systems, it is also a good word processor. Apple Writer II has some more advanced features you may need by the time you get into high school. Finally, we wrote this book using Super Text 40/56/70 (Muse). It is a fairly sophisticated word processor that can do anything up to professional level work. It is more difficult to learn, but once you have learned all of its tricks, actual word processing with Super Text is a lot easier than some of the others which may be easier to learn initially. It even gives you up to 70 columns on the screen without having to use an 80 column card. All of these are excellent word processors. When using a word processor (W/P), it is usually very easy to use and understand. This is called, "User Friendly." The word processor will allow you to do such things as realign text, move blocks of words, and easily rewrite essays.

* * * E x a m p l e * * *

This is a dictionary definition of the word *computer* : a programmable electronic device that can store, retrieve, and process data.

Definition of the word *memory:* for a computer it can be inserted and stored and from which it may be extracted when wanted; for example, 16K bytes is equal to 16384 (16 X 1024) pieces of information.

* * * After using the align function * * *

This is a dictionary definition of the word *computer* : a programmable electronic device that can store, retrieve, and process data.

Definition of the word *memory:* for a computer it can be inserted and stored and from which it may be extracted when wanted; for example, 16K bytes is equal to 16384 (16 X 1024) pieces of information.

There is sure a noticeable difference between the nonaligned definitions and the aligned definitions. Of course this is a little exaggerated, but it is to show you a point about the align function. We'll now give you a short list of functions included with most word processors:

<1> WordWrap when at the end of a line and you are still typing a word, the WordWrap function will bring the entire word down to the next line.

<2> Block Delete will delete a block of text that you may not want.

<3> Copy Block will copy a block of text and place it where indicated by the user.

<4> Line Delete deletes a single unwanted line.

<5> Find the word processor will find the character you indicate.

<6> Replace searches the entire document for a character or word, and replaces it with another one.

<7> Save saves text to tape or to disk.

<8> Load loads text from tape or disk.

<9> Append tacks on a file of text to the end of the existing file in memory. (combining letters)

<10> Print prints out the file in memory to a printer.

WRITTEN ASSIGNMENTS ON WORD PROCESSORS

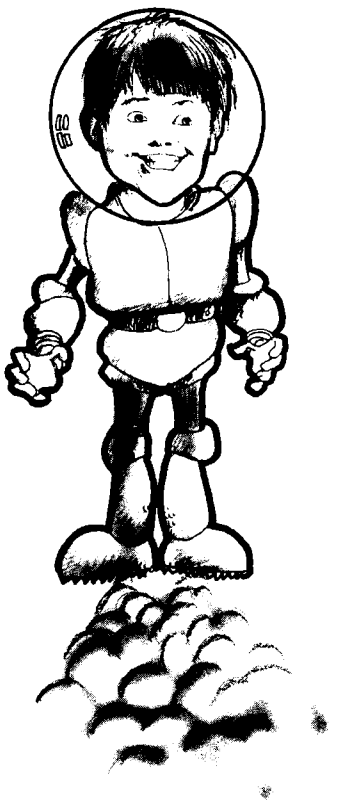
Whenever we're given a written assignment like a report or grammar homework, after we write the assignment, our parents go over it and check for correct spelling and English usage. Then we have to write the whole thing over again. Using a word processor, it's really simple. All we have to do is go back and change the mistakes, and then send it to the printer. The printer will print the assignment as many times as you want until it's correct. You don't have to re-type anything except to fix the mistakes. It helps you concentrate on correct grammar and spelling instead of all the work in re-doing the whole paper. Also, when the paper is turned in, it looks much better and clearer than a handwritten assignment.

There are many many more such commands like these which will enable you to produce great looking material. Through practice you will find that the word processor can be a very useful tool for doing just about anything.

BILLY LOST.

13

OUR FAVORITE GAMES AND PROGRAMS



Hello there. In this chapter you will not be learning any new words nor will you learn anything about programming your computer. In this chapter we want to tell you about our choices for the top ten games for you, your friends, and your family. After we finish with that we will show you some of the best programs to buy if you are interested in the different aspects of programming.

The first game is CHOPLIFTER by Broderbund Software. You pilot a helicopter that takes off from your headquarters blasting open barracks, rescuing men, and then bringing them back to your HQ (headquarters). There are three enemies that oppose you: a tank, a jet plane, and an alien in a space pod. The game ends when all the hostages in the barracks are either dead or safe back at headquarters.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

Another one is THE BILESTOAD by DATAMOST. This game is very bloody because you are a human placed on an island with heavy armor, a shield, and a double bladed axe. You can fight the robot or another one of your friends. You might get one of your arms chopped off and if you do, try to find the transporter to escape.

TUBEWAY by DATAMOST is another game for arcade action fans. You are a ship controlled by either a paddle, a joystick, or the keyboard, blasting your way in and out of different mazes. If an alien ship gets in your path on the maze, be sure you can find a line that goes all around the grid. If you can find it, hope it gets under you and then fire and everything except you will disappear from the screen.

SWASHBUCKLER by DATAMOST is a test of skill in the art of sword fighting. You are a heroic swashbuckler who tries to save himself by killing monsters and pirates, if you're good enough, you can even face the Japanese Samurai warrior. This game is a game of luck and skill, especially when the weasel or scorpion tries to kill you.

ZAXXON by Datasoft puts you on a space fighter and you blast your way over, around and through all kinds of obstacles. There are several different levels, each one more difficult than the last.

Finally, if you want a whole lot of simple but really fun arcade-like games, BEAGLE BAG by Beagle Bros. is a fantastic deal. All of the games are written in BASIC, and you can LIST them to see how they work. The little tip book that comes with the games is neat too.

Some games are more like adventures, and they require both fast reflexes and thinking. ZORK I and ZORK II by Infocom are really addictive adventure games. You are confronted with all kinds of obstacles and interesting opponents. This game requires thought and planning and will keep you up all night!

AZTEC by DATAMOST is a graphic adventure game with lots of action and interesting monsters and animals. You are an archaelogist who knows that a professor has located the Aztec ruins, but never returned. Legends say that there is an idol worth millions, and you are trying to get the idol and get out alive.

A game that talks to you in German is CASTLE WOLFENSTEIN by Muse Software. You are a prisoner of the Nazis, armed with a gun and ten bullets. You try to get past guards by either shooting them or slipping by unnoticed. By opening a chest you may be lucky enough to find a German uniform so they won't easily spot you. Don't get the SS guards on your trail. If you do, try to get a grenade from one of the chests. You'll need it.

Another really neat game by Muse is ROBOT WAR. In this game you have to program a robot in a special robot language. The language is something like BASIC and another language called Pascal. Once you have programmed your robot, it is set out on a battlefield to fight other robots that have been programmed. If your robot is programmed the best, it will survive the attacks of the other robots and blast them to "kingdom come." You can learn a lot about programming and have a good time as well.

PROGRAM PROGRAMS

Some programs help you do all sorts of things with your computer while helping you program.

These programs are called "utilities," and there are a number we can use with the Apple. The most useful is Global Program Line Editor. This program is used when you want to change a line in a BASIC program. Instead of having to rewrite the whole line if there's a mistake in it, you just used the "edit" feature of the program to change the mistakes. It is something like a word processor that you use to fix up programs. It is the most useful utility program you can have.

A program you will enjoy and learn a lot about disks with is DOS BOSS by Beagle Bros. You can put your name on the disk where the volume number goes, make invisible files names and fix up your diskette in all kinds of neat ways. It comes with a Tip Book that will teach you a lot about your disk drives and diskettes.

DOS TOOL KIT by Apple is several different utilities on the same disk. Besides having Applesoft Programmer's Aid, which has automatic line numbering and re-numbering, there are programs for graphics. Finally, THE PRINTOGRAPHER by Southwestern Data Systems is the best program for printing graphics to your printer. In the chapter on printers, we saw how to put text on a printer, but doing the same thing

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

with graphics is very difficult. PRINTOGRAPHER makes it easy, and it has a utility for saving your graphics in a very small space.

COMMUNICATIONS

How would you like to talk to another computer or send your friend a program over the telephone lines? With a modem and communications software you can do that. There are a lot of modems for the Apple. The Hayes Micromodem is a good one and so is the Novation Apple CAT. With the modem, you will need a communications program. With some modems, they give you the communications software free. If they do not, ASCII EXPRESS: THE PROFESSIONAL by Southwestern Data Systems is an excellent one. It is a little hard to learn, but once you get used to it, there's not much it can't do.

*** WANNA GO TO JAIL? ***

Some kids who have modems have been calling places that need special passwords. They break into these systems by illegally using stolen passwords. It's really easy to do once you have the password, and most of these systems are poorly

protected. Usually, there's nothing for kids in these networks, but if they get caught, the FBI or police take their computers away and make their parents pay a fine.

Since communications with computers is a lot of fun, there ought to be something for kids. Well there is, and it's free! Go to your computer store and ask them for the number of a local bulletin board. When you get the number, call up the bulletin board and leave a message for SYSOP (that's short for System Operator.) Leave your name and phone number and the secret code word you want for your own password. Then, when you get your password, you can call them up on your computer and talk with other kids (and even adults) who also have computers. Some places even have bulletin boards just for kids.

OTHER NEAT STUFF

A good way to get a lot of free programs is to join an Apple Club. Most cities have clubs for Apple users, and usually your computer store or a teacher will be able to tell you who to call. When you are a member, the club will give you, or sell

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

for a little more than the price of a diskette, a whole load of public domain software. There are a lot of neat games and other programs that clubs have. Some clubs have special groups for kids too.

There's an Apple magazine just for kids. It called *THE APPLE'S APPRENTICE*, and it has articles on all kinds of things for kids such as graphics, programs, puzzles, contests and similar features. One of the best things in the magazine is the "The Sourceror's Apprentice." It shows you how to program in machine language. (Advanced, but fast!) You can get the magazine in computer stores or for subscriptions write to:

> *THE APPLE'S APPRENTICE*
> P.O. Box 582-AA
> Santee, CA 92071

Another magazine you might like to see is *JOYSTIK*. It's a magazine that gives you tips on winning at home computer and arcade games. If you can't find it in stores, write to:

> *JOYSTIK*
> 3841 W. Oakton St.
> Skokie, IL 60076.

That's about it for our favorite games and programs and tips on some other things you might like. We like games a lot, but it's even more fun to write your own games and programs. In the next chapter we've written a bunch of programs you might like, including an arcade game. In no time at all, you'll be writing your own programs.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


PROFITS HAVE INCREASED FIFTY PERCENT SINCE WE COMPUTERIZED THE VOO DOO DIVISION!
COMPETITION
PINS

14

APPLE PROGRAMS

We've filled this chapter with several different types of programs for you to learn from and use on your Apple computer. There will be some advanced programs, but don't worry about understanding them right away. The first program fills (paints) the entire high resolution screen with the color set available (0-7). The program's intended use is for setting and fine-tuning your color set. Experiment with different color, tone, intensity, and contrast.

SCREEN PAINT

```
10 TEXT : HOME
20 HGR
30 FOR A = 0 TO 7
40 VTAB 22: PRINT "COLOR
    NUMBER=";A;" : REM TRY
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
     ADJUSTING YOUR TV OR MONITOR
     FOR DIFFERENT COLORS"
 50 HCOLOR= A
 60 HPLOT 0,0
 70 CALL 62454: REM *** FILLSCREEN ***
 80 GET A$
 90 NEXT A
100 TEXT : HOME
```

The second program draws two boxes with an opening and closing eye in the center. Different colors and patterns also appear.

ODD EYES

```
 10 TEXT : HOME
 20 C = 135
 30 HGR2
 40 FOR N = 1 TO 100
 50 A = INT (7 * ( RND (1)) + 1)
 60 B = INT (7 * ( RND (1)) + 1)
 70 C1 = INT (7 * ( RND (1)) + 1)
 80 C2 = INT (7 * ( RND (1)) + 1)
 90 FOR M = 1 TO 5
100 HCOLOR= A
110 FOR T = 0 TO 102 STEP C1
120 HPLOT T + C,0 TO 100 + C,T TO
```

```
    T + C,100 TO 0 + C,T
130 NEXT T
140 HCOLOR= B
150 FOR T = 102 TO 0 STEP - C2
160 HPLOT T,0 TO 100,T TO T,100 TO 0,T
170 NEXT T
180 FOR PAUSE = 1 TO 170
190 NEXT N
```

The third program is very interesting and fun to watch. The computer starts in the middle of the screen and draws a circle with a random color using a random rate or scale. When the program finishes the circle it re-draws itself, this time using new colors and speed variations. You should have a color set for this one!!!

```
 10 TEXT : HOME
 20 HGR2
 30 C = INT ( RND (1) * (7) + 1)
 40 B = INT ( RND (1) * (10) + 3)
 50 HCOLOR= C
 60 FOR X = 1 TO 279 STEP B
 70 HPLOT 140,96 TO X,0
 80 NEXT X
 90 FOR Y = 1 TO 191 STEP B
100 HPLOT 140,96 TO 279,Y
110 NEXT Y
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
120 FOR X = 279 TO 1 STEP - B
130 HPLOT 14Ø,96 TO X,191
14Ø NEXT X
15Ø FOR Y = 191 TO 1 STEP - B
16Ø HPLOT 14Ø,96 TO 1,Y
17Ø NEXT Y
18Ø GOTO 3Ø
```

The fourth program deals with police work! The program plays the famous tune dum-de-dum-dum dum-de-dum-dum dummmmmm.

NOTE: you must load and run the sound routine before you RUN *this program. To do so put in the sound routine found in the High Resolution/ Sound Chapter.*

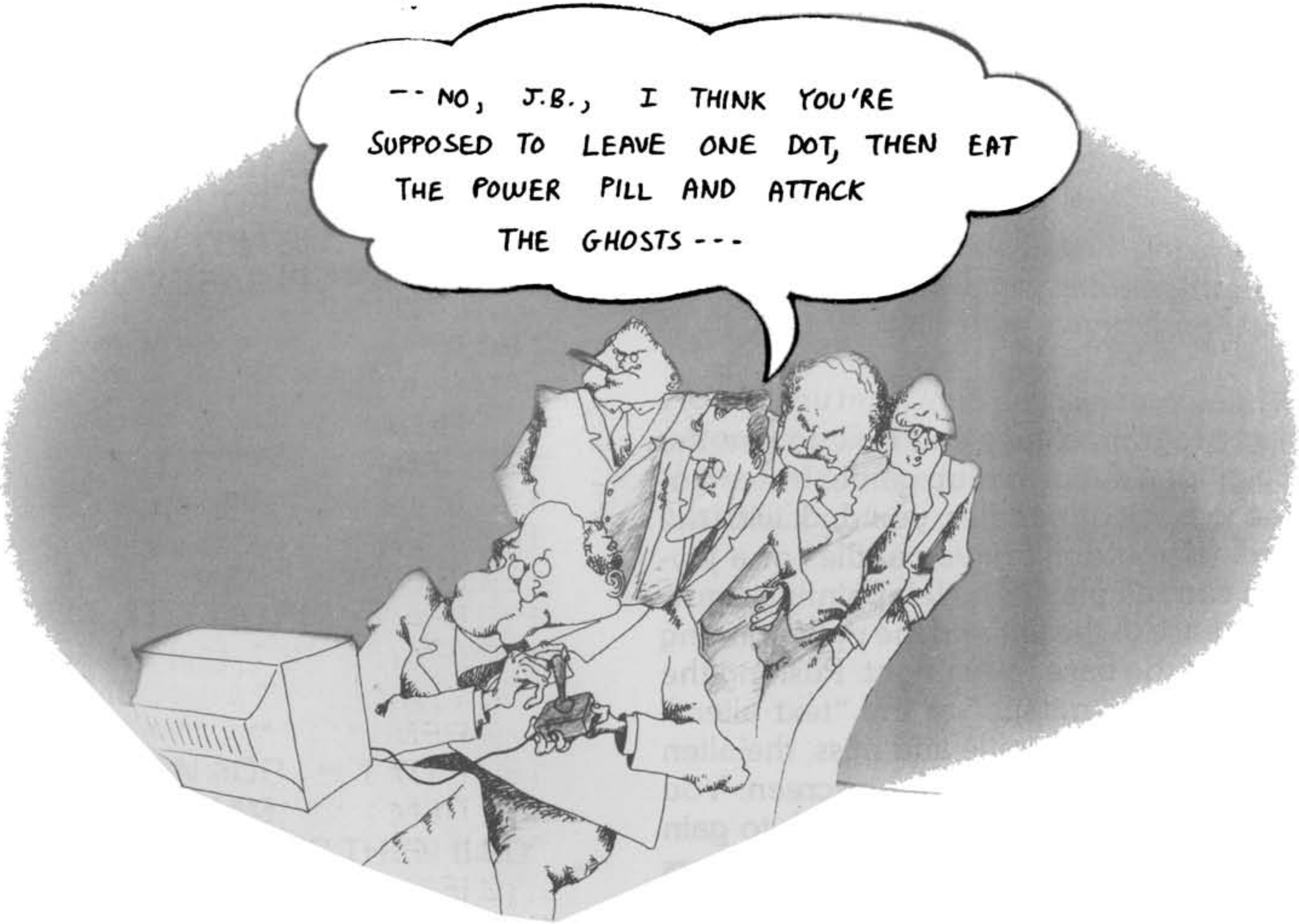
CALL THE COPS!

```
1ØØ TEXT : HOME
11Ø READ T,D
12Ø IF T = 555 THEN FOR I = 1 TO
    D: NEXT I: GOTO 11Ø
13Ø IF T = 999 AND D = 999 THEN END
14Ø POKE Ø,T
15Ø POKE 1,D
16Ø CALL 768
```

```
 170 GOTO 110
 180 DATA 84,230,555,20,103,55
 190 DATA 555,01,111,55,555,250
 200 DATA 84,88,555,500,84,230
 210 DATA 555,20,103,55,555,01
 220 DATA 111,55,555,250,84,88
 230 DATA 555,120,134,245,999,999
5000 REM *** IF SOUND ROUTINE
5010 REM *** IS NOT IN MEMORY
5020 REM *** PLEASE DO SO.
```

This next program shows you how to use the text screen for a shoot-em-up game. Instead of using the joystick or paddles, you do it with the keyboard. (If you don't have paddles or a joystick, you can still play!) The left pointing arrow moves the base to the left and the right pointing arrow moves the base to the right. Pushing the space bar fires a "missile" at the "text alien." Each time you fire a missile and miss, the alien moves closer to the bottom of the screen. You have to work quickly against the clock to gain points. When you hit the alien, the screen clears and a new alien appears at the top of the screen. See how high you can score before your time runs out or the alien lands!

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO


-- NO, J.B., I THINK YOU'RE SUPPOSED TO LEAVE ONE DOT, THEN EAT THE POWER PILL AND ATTACK THE GHOSTS ---

TEXT ALIEN

```
 10 TIME = 250:PTS = 0
 20 TEXT : HOME :FLAG = 0
 30 H = 15:V = 22:AV = 1:AH = 20
 40 REM **********************
 50 REM READ KEYBOARD
 60 REM **********************
 70 P = 0: IF PEEK ( - 16384) > 127 THEN
    P = PEEK ( - 16384) - 128
    : POKE - 16368,0
 80 REM ********************************
 90 REM RANDOM ALIEN
    HORIZONTAL MOVE
100 REM ********************************
110 FH = INT ( RND (1) * (2) + 1)
120 IF FH = 1 THEN AH = AH + 3
130 IF FH = 2 THEN AH = AH - 3
140 IF AH < 2 THEN AH = 2
150 IF AH > 38 THEN AH = 38
160 VTAB AV: HTAB AH: PRINT "=0="
170 REM ***************************
180 REM MOVE OR FIRE PLAYER
190 REM ***************************
200 IF P = 32 THEN GOSUB 400
210 IF P = 8 THEN H = H - 1
220 IF P = 21 THEN H = H + 1
230 IF H > 39 THEN H = 39
```

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

```
240 IF H < 1 THEN H = 1
250 IF V > 23 THEN V = 23
260 IF V < 1 THEN V = 1
270 HTAB H: VTAB V: PRINT "¶"
280 FOR HOLD = 1 TO 100: NEXT HOLD
290 HTAB AH: VTAB AV: PRINT " "
300 HTAB H: VTAB V: PRINT " "
310 IF FLAG = 1 THEN 20
320 REM ******************************
330 REM DISPLAY TIME AND SCORE
340 REM ******************************
350 TIME = TIME - 1: IF TIME = 0 THEN
    HOME : PRINT "FINAL SCORE =";
    PTS: END
360 HTAB 1: VTAB 23: INVERSE : PRINT
    " TIME = ";TIME: HTAB 30: VTAB 23
    : PRINT " SCORE = ";PTS: NORMAL
370 GOTO 70
400 REM ***********
410 REM FIRE!
420 REM ***********
430 FOR F = 22 TO 1 STEP - 1
440 HTAB H: VTAB F: PRINT "*"
450 HIT = (H = (AH + 1)) AND (F = AV)
460 IF HIT THEN FLAG = 1: GOTO 600
470 HTAB H: VTAB F: PRINT " ": NEXT
480 HTAB AH: VTAB AV: PRINT " "
490 AV = AV + 1: IF AV = 22 THEN
```

```
      PRINT "THEY GOT YOU!": END
500 RETURN
600 REM ******************
610 REM *** HIT ***
620 REM ******************
630 HTAB AH: VTAB AV: PRINT "BOOM!"
640 PRINT CHR$ (7)
650 FOR I = 1 TO 500: NEXT
660 AV = 1: HOME
670 PTS = PTS + 1
680 GOTO 500
```

This next program shows you a very good sort routine. A few years ago, a software company called Beagle Bros. had a contest to see who could write the fastest sort routine. The winner was a kid named Andrew Zaslow from Westport, Ct., who was a high school senior at the time. We used the sort that Andrew made in a program that will alphabetize a list of words. We weren't sure of the name of the sort routine; so we called it "Beagle Sort" after the name of the company that had the contest.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

KIDS TO KIDS TO KIDS TO

BEAGLE SORT

```
 10 TEXT : HOME
 20 VTAB 10: INPUT "NUMBER OF NAMES
    TO SORT-> ";N
 30 DIM A$(N)
 40 HOME : FOR NA = 1 TO N
 50 PRINT "NAME #";NA;"=>";:
    INPUT " ";A$(NA)
 60 NEXT NA
 70 HOME : SO$ = " SORTING "
 80 VTAB 10: HTAB 20 - LEN (SO$) / 2
    : FLASH : PRINT SO$: NORMAL
100 REM ******************
110 REM BEAGLE SORT
120 REM ******************
130 DIM B$(N): DIM A(26): DIM B(N)
140 FOR X = 1 TO N:B(X) = ASC (A$(X)) - 64
    :A(B(X)) = A(B(X)) + 1: NEXT
150 Y = 1: FOR X = 1 TO 26:E = Y:Y =
    Y + A(X):A(X) = E: NEXT
160 FOR X = 1 TO N:Y = A(B(X))
170 IF B$(Y) = "" THEN B$(Y) = A$(X)
    : NEXT : GOTO 230
180 IF B$(Y) > A$(X) THEN T$ = B$(Y)
    :B$(Y) = A$(X):A$(X) = T$
190 Y = Y + 1 : GOTO 170
200 REM ****************************
```

```
21Ø REM PRINT OUT SORTED LIST
22Ø REM ******************************
23Ø HOME : FOR X = 1 TO N:CN = CN + 1
     : IF CN = 2Ø THEN GOSUB 3ØØ
24Ø PRINT X;"."; SPC( X < 1Ø);B$(X): NEXT
25Ø END
3ØØ REM ******************
31Ø REM SCROLL STOP
32Ø REM ******************
33Ø PRINT : INVERSE : PRINT " HIT ANY
     KEY TO CONTINUE";: NORMAL : GET A$
34Ø CN = Ø: HOME : RETURN
```

Our final program is weird, but it's a lot of fun. It will invert whatever is on the screen. It changes inverse to normal and normal to inverse. It does this by scanning the screen memory and changing the value of whatever is there to its opposite. Catalog your disk, list a program or something to get text on the screen. Then RUN REVERSE and it will not clear the screen; instead it will reverse the text and background colors. When you RUN it a second time, it changes everything back to what it was originally.

REVERSE

```
 10 FOR PA = 1024 TO 1104 STEP 40
 20 FOR X = PA TO 2039 STEP 128
 30 FOR SCREEN = 0 TO 39
 40 N = PEEK (X + SCREEN)
 50 IF N > = 192 THEN F = N - 192
 60 IF N < 192 THEN F = N - 128
 70 IF N < 160 AND N > 31
    THEN F = N + 128
 80 IF N < 32 THEN F = N + 192
 90 POKE X + SCREEN,F
100 NEXT SCREEN
110 NEXT X: NEXT PA
```

Well that's it for our program chapter. We hope you had fun with the programs and maybe learned some extra tricks on your Apple. Even if you didn't understand everything, don't worry. In time you will. Try using some of the subroutines you saw in this chapter in your own programs. That will help you write programs in terms of blocks, with each routine being a block. Pretty soon, you can just put the blocks together and make any kind of program you want.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS T

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

* * * GLOSSARY * * *

ARRAY An arrangement or pattern of data in succession, such as a table of numbers. Also, a sequentially stored set of variables.

ASCII An acronym for American Standard Code for Information Interchange. It is a standard code adopted by most computers to remain compatible with others.

BASIC An acronym for Beginner's All-purpose Symbolic Instruction Code. This language is easy to learn and easy to use.

BINARY Pertains to the number system with only two digits, 1 and 0.

BOOLEAN ALGEBRA A portion of logic which is similar to algebra, but deals with logical relationships rather than numeric ones.

BRANCH Alternative directions a program can take.

BUG A mistake in a computer program, or a malfunction in the computer's hardware.

CASSETTE A means of magnetic tape storage for programs, data, and files.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

CELL A storage place in the part of memory with the capacity to hold only one bit.

CENTRAL PROCESSING UNIT The main component of any computer, the brain of the operations. The CPU on the Apple is a 6502 microprocessor.

CHARACTER Any letter, number, or symbol stored by the computer.

COBOL An acronym for COmmon Business Oriented Language. A high level language for the development of complex business programs.

COMPUTER A device that executes mathematical or logical operations without the help of humans.

COMPUTER OPERATOR A person who knows the programming language and is able to operate peripheral devices.

COMPUTER UTILITY A program or device that allows the user to gain a better use of the computer's functions.

CRT An acronym for Cathode Ray Tube. The computer's TV or monitor is the CRT.

DATA Information that the computer can accept. This can include both the computer program (software) and the information that the computer processes. Also a statement in BASIC.

DEBUG To find and fix a problem that might exist in a program.

DECIMAL DIGIT A numbering system that is base 10. Numbers include 0, 1, 2, 3, 4, 5, 7, 8, 9.

DELETE To eliminate or remove a file name; the contents of the file might remain.

DIGITAL COMPUTER A device that operates on a base of ones and zeros.

DISK DRIVE A form of storage for programs, files, data using hard or floppy disks or diskettes.

DISKETTE The media for the disk drive, sometimes referred to as the disk.

DOS An acronym for Disk Operating System.

EDIT To check, change or insert data into a program or file.

ERROR A problem or bug in a program usually caused when the computer cannot process the information or command given.

FILES Programs or data that are stored on tape or diskette and can be called up again for later use.

FIRMWARE Software that is permanently stored in the computer. Storage media is ROM (Read Only Memory).

FLOWCHART A diagram that shows the logical operation of a program through the use of various symbols.

FORTH A high level fast language that uses user-defined words to create programs. This language is available for the Apple.

FORTRAN An acronym for FORmula TRANslator. A high level language used primarily for making highly complex scientific and engineering computations.

GLITCH An unexplained accidental loss of a piece of data.

GRAPHICS A mode that allows the computer to form colorful or complex visual displays and drawings.

HARD COPY A printed copy of the output of a program, listing, or graphic display.

HARDWARE The physical part of the computer. Keyboard, CPU, RAM chips, ROM chips, etc.

HEXADECIMAL A numbering system that is on the base of 16. Digits include 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

HOME COMPUTER A small low cost computer that is generally from $100 to $1000. We think they were invented so that kids could have a computer in their homes.

INFORMATION The flow of data from one point to another.

INPUT Inserting information into the computer.

INPUT/OUTPUT Means of sending and receiving information such a Disk Drive or Cassette Deck. Usually abbreviated as I/O.

INTEGRATED CIRCUIT A complex complete circuit that requires minimal parts to get a desired circuit operation such as a computer.

I/O An abbreviation for Input Output.

K An abbreviation for kilo or 1000. (Actually refers to 1024, 2 to the 10th).

KEYBOARD A group a keys manually operated for inputting information into the computer.

KEYWORD A major word element in a particular programming language. In BASIC, such words as RUN, FOR, NEXT, GOTO, PRINT are keywords.

LIGHT PEN An electronic device that allows input to the computer by the use of light or darkness on the CRT screen.

LOAD To read in information from an external device such as a Disk Drive and/or Cassette Deck.

MEMORY The part of the computer that allows storage of programs and data.

KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO KIDS TO

MICROCOMPUTER A small, low cost computer that is generally used for home, small businesses, and especially made for kids.

PASCAL A highly structured programming language originally used to help students learn structured programming.

PROGRAM A set of instructions that guides and informs the computer exactly what the user wants to do.

RAM An acronym for Random Access Memory. The process of obtaining data from or placing data into storage (memory).

RUN A single statement for executing a program.

SAVE To save information to an external device such as a diskette and/or Cassette Deck.

STRING A variable that stores any kind of characters generated by the computer in ASCII form.

SURGE A sudden jump in the AC line voltage or other source voltages that may cause unexpected wipe outs of computer memory.

SYNTAX The grammatical and base structural laws of a language.

WHOLE NUMBER A number without any fractional parts; e.g., 12, 45, 54, 99 , not 1 2/3, 4/5. Also referred to as an INTEGER.

WORD A complete word is comprised of 8 bits.

FOR THE APPLE COMPUTER

Written by kids for kids, this unique book explains Applesoft BASIC programming on the Apple II, II+ and //e. Created from the idea that kids can teach other kids better than anyone else, the material is designed to help you get started using and programming your Apple Computer.

You'll learn how to use the disk drive, PRINT and math statements, variables, loops, branching and subroutines, and arrays. Two chapters are devoted to sound and graphics and another will teach you how to write an original game. Before long, you'll be using your Apple Computer to finish your homework in record time!

As an added bonus, the authors discuss their favorite programs and games. Complete with a computer glossary, KIDS TO KIDS ON THE APPLE COMPUTER will teach you the magic of programming in simple, straightforward language.

COMING SOON FOR THE COMMODORE 64

OTHER POPULAR COMPUTER BOOKS BY DATAMOST:

Compumath Magic
Computer Playground Apple, II, II+ & //e
Computer Playgound Atari 400/800/1200
Computer Playground Commodore 64/VIC-20
Computer Playground TI-99/4A
by M.J. Winter

Games Apples Play
Games Ataris Play
Games TIs Play
by Mike Weinstock & Mark Capella

The Atari Experience
by Adrien Z. Lamothe
The Commodore Experience
by Mike Dean Klein

Apple Almanac
The Elementary Apple
The Elementary Atari
The Elementary Commodore 64
The Elementary IBM-PC
The Elementary Timex/Sinclair
The Elementary VIC-20
The Elementary TI-99/4A
by William Sanders

How to Write an Apple Program
How to Write an IBM-PC Program
How to Write a TRS-80 Program
How to Write a Program Vol. II
Computer in Your Pocket
by Ed Faulk

Kids to Kids on the Color Computer
by Billy Sanders & Sam Edge

Kids & the Apple
Kids & the Atari
Kids & the Commodore 64
Kids & the IBM-PC/PCjr
Kids & the Panasonic
Kids & the TI-99/4A
Kids & the VIC-20
by Ed Carlson

Using 6502 Assembly Language
p-Source
by Randy Hyde

DATAMOST™
8943 Fullbright Avenue, Chatsworth, CA 91311-2750
(818) 709-1202

ISBN 0-88190-316-7

0 48831 00031